VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Vyhledávání síťových anomálií pomocí síťových toků a systémových informací

Network Anomaly Searching with Network Flow and System Informations

2015

Bc. Dalibor Zegzulka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Dalibor Zegzulka**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Vyhledávání síťových anomálií pomocí síťových toků a systémových informací
Network Anomaly Searching with Network Flow and System Informations

Zásady pro vypracování:

Cílem diplomové práce je navrhnout systém pro vyhledávání síťových anomálií na základě síťového provozu a systémových informací získaných z různých zařízení.
1. Typy síťových útoků a anomálií.
2. Popis formátu souboru PCAP.
3. Sběr síťových a systémových informací.
4. Vyhledávání anomálií na základě klasifikace získaných dat.
5. Návrh IDS/IPS systému, založeného na anomáliích.
6. Ověření funkčnosti navrženého IDS/IPS systému.

Seznam doporučené odborné literatury:

Roberto Pietro, Luigi V. Mancini *Intrusion Detection Systems (Advances in Information Security)* Springer 2010, ISBN-13: 978-1441945853
Ryan Trost *Practical Intrusion Analysis: Prevention and Detection for the Twenty-First Century: Prevention and Detection for the Twenty-First Century* Addison-Wesley Professional 2009, ISBN-13: 978-0321591807
Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Nevlud**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 21. dubna 2015 .....

Chtěl bych velmi poděkovat svému vedoucímu práce Ing. Pavlu Nevludovi za jeho cenné rady a odbornou pomoc při realizaci této diplomové práce. Dále bych chtěl poděkovat všem svým blízkým za podporu během celého mého studia na vysoké škole.

Abstrakt

Tato diplomová práce se zabývá vytvořením monitorovacího systému pro vyhledávání síťových anomálií na základě síťového provozu a systémových informací získaných z různých zařízení. Vytvořený systém je založený na open-source nástroji RRDtool a na protokolu SNMP. Pro vizualizaci výsledků byl použit webový server. Nástroj RRDtool slouží pro ukládání časově závislých dat a má v sobě také implementovanou metodu Holt-Winters, která slouží pro predikci vývoje časové řady. Pro práci s tímto nástrojem jsem použil skriptovací jazyk Perl. Skripty, které jsem vytvořil, tvoří jádro celého monitorovacího systému pro detekci anomálií. Vytvořený systém patří do kategorie IDS. Funkčnost navrženého systému byla otestována v programu GNS3, v laboratoři počítačových sítí a na zařízení, které bylo připojené přímo k internetu.

Klíčová slova: RRDtool, SNMP, Perl, skripty, detekce anomálií.

Abstract

This master thesis is about creation of monitoring system for searching network anomalies based on network traffic and system information obtained from different devices. Created system is based on open-source tool RRDtool and on SNMP protocol. For visualization of results was used web server. RRDtool serves for saving time-dependent data and it has also implemented Holt-Winters method for prediction of evolution of time series. I used scripting language Perl for working with this tool. Scripts, which I've made, form the core of entire monitoring system for anomaly detection. Created system belongs to IDS category. Functionality of designed system was tested in GNS3 software, in laboratory of computer networks and on device, which was connected directly to internet.

Keywords: RRDtool, SNMP, Perl, scripts, anomaly detection.

Seznam použitých zkratek a symbolů

ARP	–	Address Resolution Protocol
CF	–	Consolidation Function
DHCP	–	Dynamic Host Configuration Protocol
DoS/DDoS	–	Denial of Service/Distributed Denial of Service
DS	–	Data Source
DNS	–	Domain Name System
DTS	–	Data Source Type
EUI-64	–	64-bit Extended Unique Identifier
ICMP	–	Internet Control Message Protocol
IDS	–	Intrusion Detection System
IPS	–	Intrusion Prevention System
IPv4	–	Internet Protocol version 4
IPv6	–	Internet Protocol version 6
MIB	–	Management Information Base
MLD	–	Multicast Listener Discovery
MTU	–	Maximum Transmission Unit
OID	–	Object Identifier
PDU	–	Protocol Data Unit
PDP	–	Primary Data Point
RRA	–	Round-Robin Archive
RRDs	–	Round-Robin Database
RRDtool	–	Round-Robin Database tool
SNMP	–	Simple Network Management Protocol
SSH	–	Secure Shell
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
URL	–	Uniform Resource Locator
UTC	–	Coordinated Universal Time
VoIP	–	Voice over Internet Protocol
VPN	–	Virtual Private Network
XML	–	Extensible Markup Language

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V dnešní době se sítě staly součástí každodenního života a s tím roste i závislost společnosti na nich. Zasahují stále do více odvětví a oborů. Jejich velký rozvoj je spojen s celkovým rozvojem IT odvětví. Nové technologie jsou stále dostupnější i pro širší veřejnost, nejen pro oblast velkých společností. Sítě nabízejí stále více služeb a možností jejich uplatnění. To přináší, ale i jistá negativa. S rostoucí mobilitou a počtem zařízení je třeba dávat velký důraz na zabezpečení systémů a celkovou bezpečnost infrastruktury. K tomu slouží nejen monitorovací systémy, ale především bezpečnostní systémy. Tyto systémy mají stále více klíčovou úlohu ve všech společnostech. Útoky se stávají každodenní záležitostí a jejich počet roste. Proto systémy, které dokáží odhalit útoky a zejména ty neznámé, jsou velmi potřebné. Největší slabinou všech bezpečnostních systémů, ale stále zůstává lidský faktor. Další slabinou je, že mnoho firem provozuje zabezpečení pouze na úrovní IPv4 a podceňuje zabezpečení na úrovni IPv6. Zabezpečení jen na úrovni IPv4 je ve většině případů nedostačující. Pokud není zabezpečení i na úrovní IPv6, mohou být útoky vedeny právě přes IPv6 a zabezpečení na IPv4 se stává zbytečným.

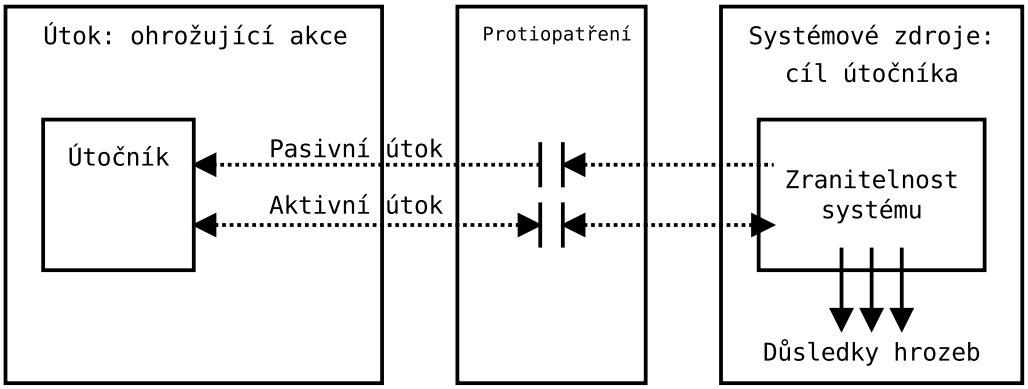
Počítačové útoky už dávno nejsou záležitostí jen jednotlivců. Vznikají organizované skupiny, které si své oběti vybírají cíleně. Nejedná se pouze o zábavu, při které se útočníci snaží překonat bezpečnostní systém a upozornit na bezpečnostní chyby. Cílem těchto útoků bývají klíčové služby dané společnosti a jejich znepřístupnění. K vyřazení nějaké služby, není nutné provádět útok přímo sám. Dnes je možné si na internetu takovéto útoky objednat. Tyto útoky se staly výnosným byznysem. Jejich cena začíná na pár dolarech a může překročit hranici tisíce dolarů. Záleží na rozsahu daného útoku. Útoky nemusí mít za cíl pouze znepřístupnění nějaké služby. V některých zemích se počítačové útoky staly součástí propagandy různých hnutí nebo politických stran. Některé vlády přímo financují a podporují tyto organizované skupiny. Útoky na této úrovni mohou dospět až do fáze kybernetické války. Z tohoto důvodu také vznikl projekt FENIX, který má chránit část české internetové infrastruktury před velkými DDoS (Distributed Denial of Service) útoky. Vznik tohoto projektu inicializoval internetový uzel NIX.CZ. Hlavní myšlenka spočívá v tom, že při útoku se síť odpojí od zbytku světa a data se budou vyměňovat jen v rámci propojených sítí od velkých tuzemských provozovatelů. [3]

Cílem této diplomové práce je popsat síťové útoky, anomálie a také popsat formát souboru PCAP. Hlavním cílem této práce je navrhnout a vytvořit monitorovací systém, který bude založený na detekci anomálií v počítačových sítích. Funkčnost tohoto systému bude otestována v laboratoři počítačových sítí. Monitorovací systém bude využívat nástroj RRDtool a protokol SNMP (Simple Network Managment Protocol). Systém se bude skládat z vlastně implementovaných skriptů v jazyce Perl. Tyto skripty budou sloužit pro práci s nástrojem RRDtool a sběr informací. Jejich obsahem bude vytvoření databáze, naplnění databáze daty a vytvoření grafů, ve kterých se budou vykreslovat anomálie. Pro sběr síťových a systémových informací byl vybrán protokol SNMP. Pro lepší vizualizaci budou výsledné grafy umístěny na webový server. To zajistí větší přehled nad monitorovanou sítí. Součástí této práce je také ukázkový skript, který slouží pro jednoduché vytvoření vlastního skriptu na detekci anomálií.

Text této diplomové práce je rozdělen do devíti kapitol, které tvoří tři základní bloky. První blok kapitol je věnován popisu útoků a anomálií v počítačových sítích. Nejprve je popsáno základní rozdělení těchto útoků a poté následují dvě kapitoly věnující se podrobnějšímu popisu jednotlivých útoků. Jedna kapitola popisuje pasivní útoky typu *Port Scanning* a odposlouchávání. V další kapitole jsou popsány aktivní útoky typu: *Denial of Service/Distributed Denial of Service*, *Spoofing Attack*, *Malware*, *Google Hacking* a útoky zaměřené na IPv6. Následuje blok kapitol, který se zabývá sběrem síťových a systémových informací a následným vyhledáváním anomálií v těchto datech. Jedna z kapitol se tedy věnuje sběru informací zejména pomocí protokolu SNMP. Druhá z těchto kapitol popisuje možnosti nástroje RRDtool a popis parametrů, které slouží pro detekci anomálií. Poslední blok kapitol je věnován návrhu monitorovacího systému založeného na detekci anomálii a jeho následnému testování. Nejprve je popsáno, na jakém principu celý systém pracuje a jak zprovoznit tento systém. Následně je popsáno, jak byl systém testován a výsledky těchto testů. Celá tato diplomová práce je napsána v LaTeXu. [1]

2 Základní rozdělení síťových útoků a anomálií

V této kapitole budou popsány anomálie a útoky v počítačových sítích. Síťový útok je obvykle definován jako zásah do síťové infrastruktury (viz obrázek 1). Pomocí kterého útočník nejprve analyzuje prostředí dané sítě a shromažďuje informace s cílem nalézt otevřené porty a slabá místa (neoprávněný přístup ke zdrojům) a poté zaútočit na systémové zdroje. Útoky mohou být prováděny na různých vrstvách: IP útoky, ICMP útoky, TCP útoky, útoky na směrování, útoky na aplikační vrstvě. [4]



Obrázek 1: Diagram útoku

Definice 2.1 *Úmyslný čin, kterým se subjekt snaží vyhnout bezpečnostním službám a porušit bezpečnostní politiky systému. Skutečný útok na bezpečnost systému pochází z inteligentní hrozby. [4]*

2.1 Vnitřní/Vnější útoky

Útok může být proveden mimo organizaci neoprávněnou osobou (vnější útok) nebo uvnitř organizace prostřednictvím osoby, která má přístup k síti (vnitřní útok). Vnitřní útok můžeme dále rozdělit na záměrný útok a nezáměrný útok. Záměrný útok je prováděn osobou pomocí úmyslného odposlouchávání, krádeže, poškozování informací, využívání informací podvodným způsobem nebo k odepření přístupu ostatním legitimním uživatelům. Nezáměrný útok obvykle vyplývá z nedbalosti, neznalosti nebo z úmyslného obcházení zabezpečení. [4, 5]

2.2 Pasivní útoky

Pasivním útokem útočník sleduje nešifrovaný provoz, hledá hesla v čisté podobě (cleartext) a jiné citlivé informace, které mohou být využity při jiném typu útoku. Nedochází k ovlivnění systémových zdrojů nebo k jejich omezení. Pasivní odposlech provozu sítě umožňuje útočníkovi pochopit chování v dané síti. Výsledkem pasivního útoku je získání informací nebo datových souborů bez souhlasu nebo vědomí oběti. [2]

Pasivní útoky zahrnují:

- analýzu provozu
- sledování nechráněné komunikace
- dešifrování slabě šifrované komunikace
- zachycení informací pro autentizaci

2.3 Aktivní útoky

Pomocí aktivního útoku se útočník snaží obejít nebo prolomit zabezpečení systému. Při tomto typu útoku mohou být využity viry, červi nebo trojští koně. Dochází k nepovolenému přístupu k systémovým zdrojům a jejich změně, omezení, ovlivnění nebo zničení. Výsledkem aktivního útoku je zveřejnění nebo šíření zabezpečených informací, znepřístupnění dané služby nebo úpravě dat. [2]

Aktivní útoky zahrnují:

- obcházení nebo prolomení ochranné funkce
- zavádění škodlivého kódu
- získání a úpravu informací

2.4 Anomálie

Za anomálii označujeme chování nebo jev, který neodpovídá nastaveným pravidlům. Nemusí se vždy jednat o počítačový útok. Anomálie může být způsobena aktualizací systému, špatným nastavením nebo nedostatečným výkonem. V takovém případě tuto anomálii nazýváme *False Positive*. Pokud dojde k opravdovému útoku na počítačovou síť, nazýváme tuto anomálii jako *True positive*.

Pro detekci anomálií musíme nejprve znát model chování v dané síti. Za anomálii považujeme takové chování, které neodpovídá modelu dané sítě. Výhoda systémů založených na detekci anomálií spočívá v jejich dynamičnosti. Dokáží se přizpůsobit chování v dané síti. Většina systémů je založena na porovnávání signatur (pravidel). Tyto pravidla mají pevně dané hranice a po překročení těchto hranic systém zahlásí význačnou událost. Jedná se tedy o statický model. Nevýhoda systémů založena na anomáliích je především v generování poměrně mnoha falešných poplachů (*False Positive*). [6]

Velmi důležitá je také vizualizace anomálií. Slouží k rychlejšímu a přehlednějšímu zjištění, kde problém nastal nebo co všechno bylo ovlivněno. Administrátor dané sítě nemusí procházet tak velké množství logů. Může se zaměřit na konkrétní věci. Vizualizace přináší i další výhodu v podobě samotného monitorování a přehledu o stavu sítě a síťových zařízení.

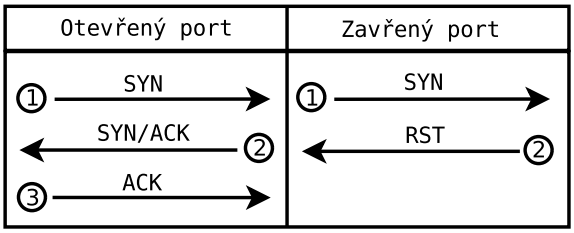
3 Pasivní útoky

3.1 Port Scanning

Jedná se o typ útoku, kdy útočník pošle několik žádostí na daný rozsah portů. Útočník zjišťuje, v jakém stavu se nachází dané TCP a UDP porty. Hledá porty na zařízení oběti, které jsou používané síťovými službami a zda jsou otevřené nebo zavřené. Útočník tímto způsobem zjistí, které typy útoků budou na tento systém účinné a které účinné nebudou. Dále pak z těchto informací může útočník zjistit, jaký operační systém běží na zařízení oběti. Tím je sice dále omezen počet útoků, které může útočník použít, ale může využít zranitelná místa charakteristická pro daný operační systém. Port Scanning využívají také bezpečnostní technici k ověření zabezpečení sítě. Používají se k tomu softwarové nebo hardwarové nástroje. [5]

3.1.1 TCP Connect scanning

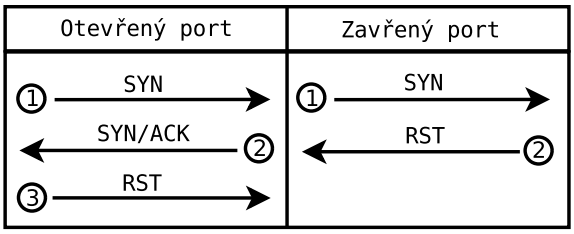
Základní technika skenování portů. Útočník testuje každý port systémovým voláním metody *Connect*. Pokud dojde k připojení na daný port, znamená to, že port je otevřený a naslouchá. Jestliže je port zavřený (nedostupný), je vracen paket s příznakem RST (viz obrázek 2). Výhodou této techniky je, že nejsou potřeba žádné zvláštní oprávnění a je velmi rychlá (skenuje se mnoho portů paralelně). Nevýhodou je snadné rozpoznání této techniky na zařízení oběti. [7, 8]



Obrázek 2: TCP Connect scanning

3.1.2 TCP SYN scanning

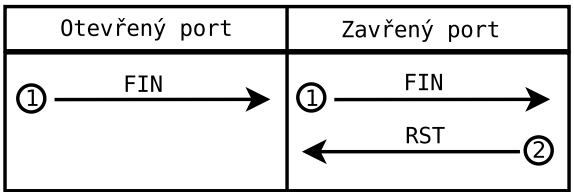
Tato technika je nazývána také jako „half-open scanning". Nedochází totiž k úplnému sestavení TCP spojení. Útočník posílá paket s nastaveným příznakem SYN (snaha navázat spojení). Zařízení oběti odpoví paketem s příznakem SYN/ACK (daný port je otevřený a naslouchá) nebo odpoví paketem s příznakem RST (daný port je zavřený). Útočník poté ihned pošle paket s nastaveným příznakem RST (ukončení spojení) (viz obrázek 3). Výhodou této techniky je, že mnohem menší počet zařízení tuto událost vloží do logu. Pro tuto techniku je nutné mít administrátorská práva na zařízení odkud je prováděn útok. To může být označeno jako nevýhoda této techniky. [7, 8]



Obrázek 3: TCP SYN scanning

3.1.3 TCP FIN scanning

Předchozí technika TCP SYN scanning není dostatečně utajena a může být uvedená v logu. Některé firewally, paketové filtry nebo IDS registrují pakety s nastaveným příznakem SYN. Pakety s příznakem FIN nemusí být zaregistrovány bezpečnostními systémy. Hlavní myšlenka spočívá v tom, že zavřené porty na paket s příznakem FIN odpoví pomocí paketu s příznakem RST (viz obrázek 4). Otevřené porty pakety FIN ignorují. Toto chování ale není pravidlem a některé operační systémy (Microsoft) odpovídají paketem RST i u otevřeného portu. Z toho plynou výhody i nevýhody této techniky. [8]



Obrázek 4: TCP FIN scanning

3.1.4 Fragmentation scanning

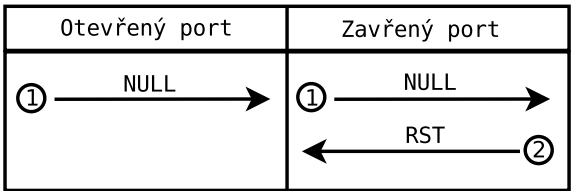
Tato technika je jen úprava předchozích technik. Místo odeslání celého paketu, se paket rozdělí na několik menších fragmentů. Při takovémto rozdělení TCP hlavičky je pro bezpečnostní systémy obtížnější útok odhalit. Tato technika nefunguje u paketových filtrů nebo firewallu, které fragmentované pakety řadí do front. [7]

3.1.5 TCP reverse ident scanning

Využívá *ident* protokol pomocí kterého, je možné zjistit uživatelské jméno, pod kterým běží proces na daném portu. Útočník může tedy zjistit, zda služba bežící na otevřeném portu má administrátorská práva. Tato technika je možná pouze při úplném TCP spojení. [7]

3.1.6 TCP Null scanning

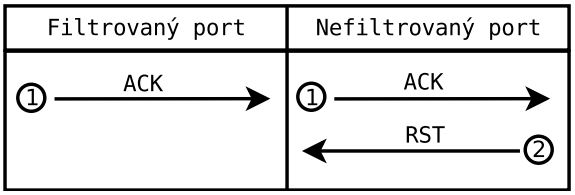
V této technice útočník posílá paket, který nemá nastavený žádný příznak v TCP hlavičce. Porty, které nejsou otevřené, odpoví paketem RST (viz obrázek 5). Porty, které jsou otevřené, tento paket ignorují a neodpoví na něj. Toto chování ale není pravidlem a některé operační systémy (Microsoft) odpovídají paketem RST i u otevřeného portu. [8]



Obrázek 5: TCP Null scanning

3.1.7 TCP ACK scannnig

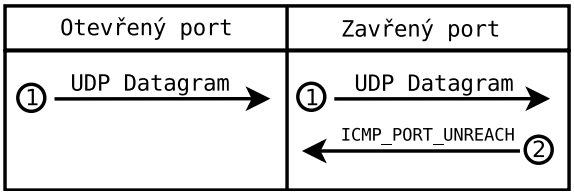
Pomocí této techniky nemůžeme určit, zda je port otevřený nebo zavřený. Útočník může zjistit pouze, zda je port filtrovaný či nikoliv. Jestliže útočník pošle paket a vrátí se odpověď v podobě paketu RST, je port nefiltrovaný. Pokud se žádná odpověď nevrátí, je port filtrovaný (viz obrázek 6). Tímto způsobem útočník zjistí nastavení firewallu. [8]



Obrázek 6: TCP ACK scanning

3.1.8 UDP ICMP port unreachable scanning

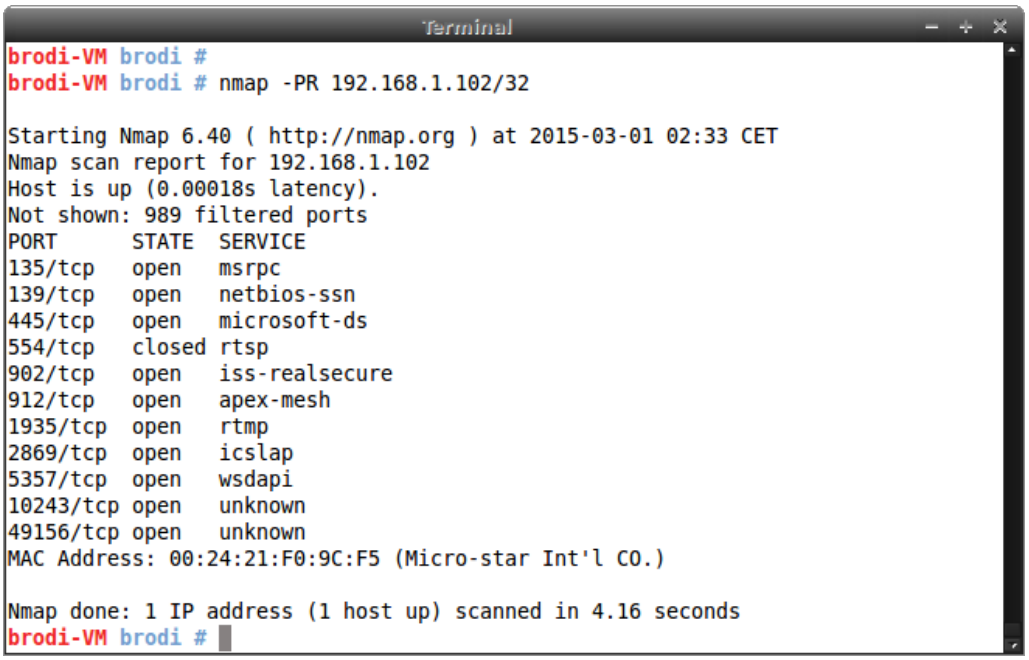
Tato technika se zaměřuje na skenování UDP portů. Přestože UDP protokol je jednodušší, jeho skenování je obtížnější. UDP protokol neposílá potvrzení přijetí jako TCP protokol. Pokud útočník pošle UDP paket na zavřený port, dostane odpověď *ICMP-PORT-UNREACH* (viz obrázek 7). Můžeme tedy zjistit, který port je zavřený, Vylučovací metodou, pak zjistíme i porty, které jsou otevřené. Tato technika je dost pomalá. Z důvodu nízké frekvence odpovědí *ICMP-PORT-UNREACH*. Je potřeba taky brát ohled na to, že *ICMP-PORT-UNREACH* zpráva nemusí být doručena podle základního chování UDP protokolu. [7]



Obrázek 7: UDP ICMP port unreachable scanning

3.1.9 Nmap

Jedním z nástrojů pro skenování sítě je open-source nástroj Nmap. Tento nástroj byl vyvinut pro rychlé skenování velkých sítí. Nmap posílá na hostitelská zařízení speciálně upravené pakety. Následnou analýzou dokáže zjistit: jaké služby nabízí toto hostitelské zařízení, na jaké verzi operačního systému běží, jaký typ paketového filtru nebo firewallu je využíván a mnoho dalších užitečných věcí. Tento nástroj je běžně používán ke kontrole zabezpečení sítě. Nmap dělí porty do 6 stavů: otevřený, zavřený, filtrovaný, nefiltrovaný, otevřený/filtrovaný a zavřený/filtrovaný. [7]

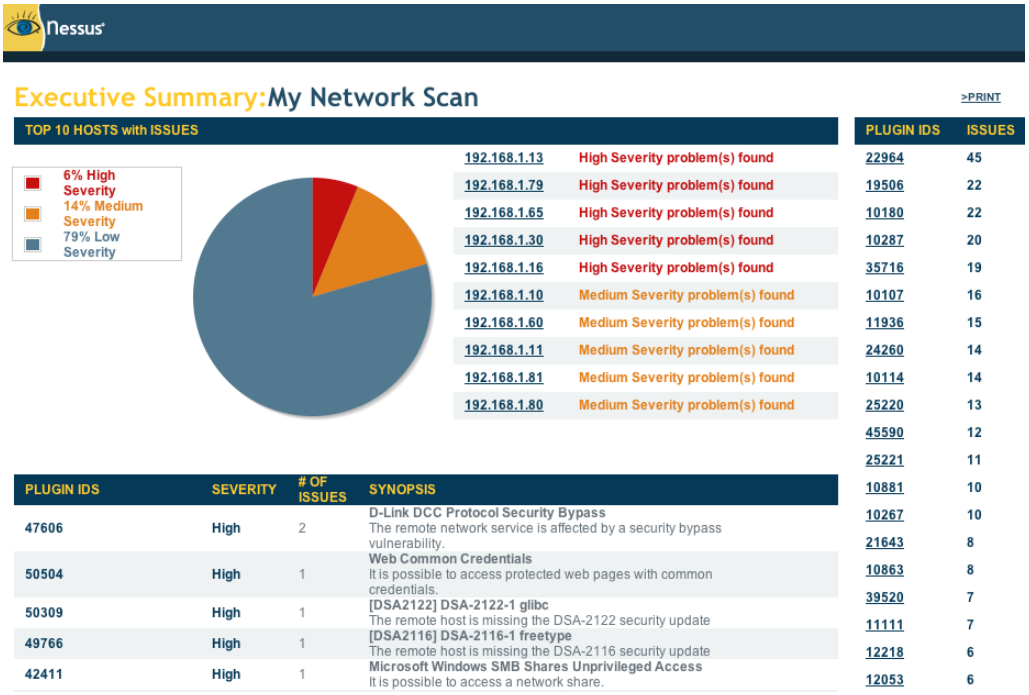


Obrázek 8: Ukázka z programu Nmap

3.1.10 Nessus

Nessus vulnerability scanner patří k nejvyspělejším skenovacím softwarům na světě. Dokáže provádět vysokorychlostní sledování a skenování jednotlivých zařízení v síti nebo

celých síťových infrastruktur. Nessus má velké množství přednastavených testů. Pomocí kterých je administrátor schopen ověřit zranitelnost jednotlivých zařízení před případným útokem. Pro nekomerční použití je Nessus nabízen zdarma.



Obrázek 9: Ukázka z programu Nessus [9]

3.2 Odposlouchávání

Pomocí techniky odposlouchávání je možné zachytit nešifrovaný provoz a následně provézt jeho analýzu. Z této analýzy můžeme zjistit velmi mnoho informací: přihlašovací údaje uživatele, hesla síťových zařízení, směrovací informace, adresy zařízení atd. Odposlouchávání je možné jen tehdy, pokud útočník má přístup na komunikační medium. Největší slabinou je tedy rozbočovač (hub), který rozesílá provoz na všechny své porty. Tímto způsobem je umožněno útočníkovi zachytit veškerou komunikaci, která se uskuteční v rámci rozbočovače. Zachycený provoz se nejčastěji ukládá do souboru pcap. Nejvyužívanějším nástrojem v této kategorii je *Wireshark*, dále pak *tcpdump* a *TShark*. [5]

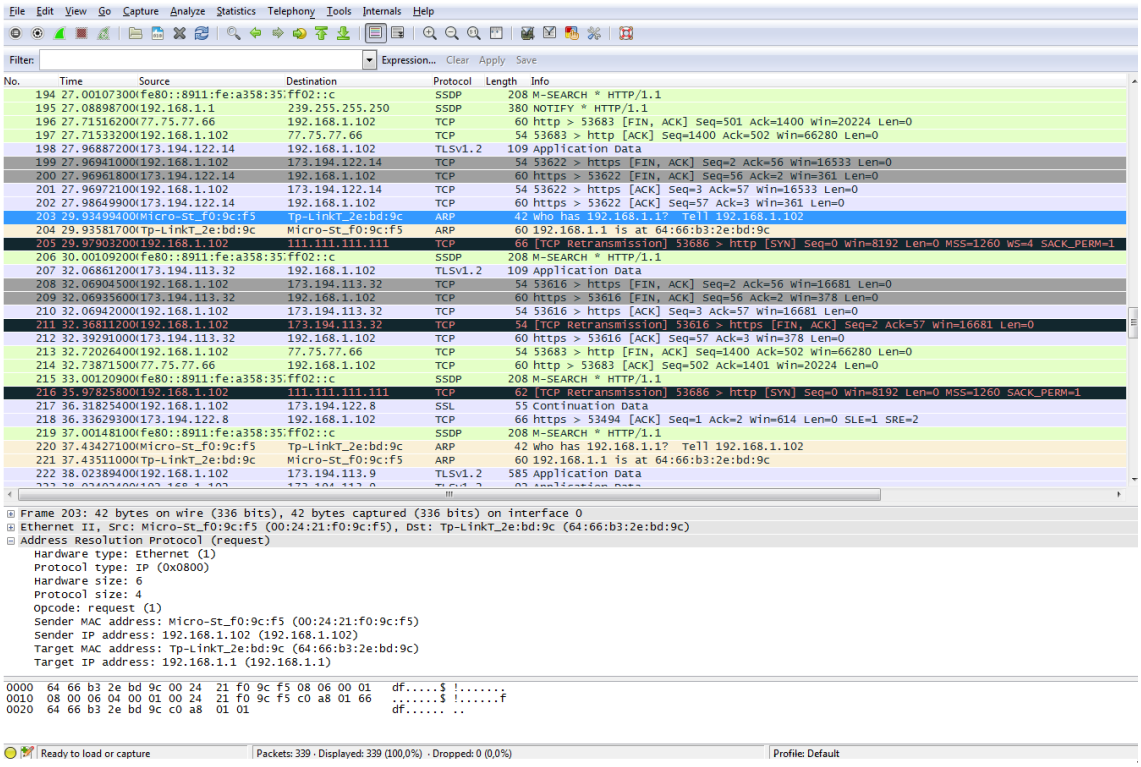
3.2.1 Wireshark

Wireshark je „open source"nástroj, šířen pod licencí GNU-GPL. Jedná se o síťový paketový analyzátor, pomocí kterého je možné zobrazit zachycená data v paketu v co nejdetailnější formě. Tento nástroj pracuje pasivně a neodesílá žádné pakety do dané sítě. *Wireshark* je využíván: síťovými administrátory při řešení problému v dané síti, bezpečnost-

ními techniky k odhalení hrozeb, vývojáři k odhalení chyb v implementaci protokolu nebo lidmi, kteří chtějí daný síťový protokol prozkoumat podrobněji. [10]

Některé z vlastností nástroje *Wireshark*:

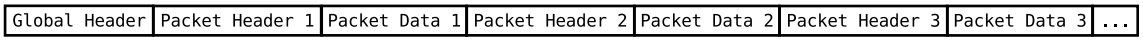
- dostupný pro UNIX i Windows
- zachycení paketů v reálném čase
- možnost otevření souboru se zachyceným provozem z jiných programů (tcpdump)
- zobrazení paketů s velmi detailními informacemi daného protokolu
- uložení zachyceného provozu do souboru
- filtrace paketů podle mnoha kriterií
- vyhledávání paketů podle mnoha kriterií
- barevné rozlišení protokolů



Obrázek 10: Ukázka výpisu z nástroje Wireshark

3.2.2 Formát souboru pcap

Jedná se o typ souboru, který patří k nejvíce využívaným pro ukládání zachyceného síťového provozu v reálném čase. Je součástí knihoven *libpcap/WinPcap* a má velmi jednoduchou strukturu. Soubor typu pcap je neoficiálním standardem pro ukládání síťového provozu na systémech UNIX. Aktuální verze formátu je 2.4. Každý soubor obsahuje globální záhlaví. To obsahuje globální informace a za ním následují záznamy o zachycených paketech. Soubor používá příponu *.pcap*. Programy, které využívají knihovny *libpcap/WinPcap*: tcpdump, ngrep, Wireshark, Snort, Nmap, Bro IDS, Suricata, Scapy a další.



Obrázek 11: Formát pcap souboru

Zachycený paket v souboru nemusí nutně obsahovat všechny data v paketu, které byly přenášeny sítí. Soubor může obsahovat nejvýše prvních N bajtů každého paketu, pro nějakou hodnotu N. Hodnota N je také nazývána jako „*snapshot length*"nebo „*snaplen*". Hodnota N by měla být větší než největší možná délka daného paketu. To zajistí, že žádný paket nebude zkrácen. Většinou je tato hodnota nastavena na 65535.

Globální záhlaví - je na začátku každého pcap souboru a je následováno záhlavím prvního paketu:

```
typedef struct pcap_hdr_s {
        guint32 magic_number; /* magic number */
        guint16 version_major; /* major version number */
        guint16 version_minor; /* minor version number */
        gint32   thiszone;     /* GMT to local correction */
        guint32 sigfigs ;      /* accuracy of timestamps */
        guint32 snaplen;       /* max length of captured packets, in octets */
        guint32 network;       /* data link type */
} pcap_hdr_t;
```

Výpis 1: Syntaxe globálního záhlaví souboru pcap

- *magic_number* - slouží k určení, v jakém pořadí budou bajty zpracovány. Pokud aplikace zapíše do tohoto pole hodnotu 0xa1b2c3d4. Jedná se o nativní řazení. Aplikace, která toto pole čte, ho může přečíst dvěma způsoby. První způsob je, že přečte hodnotu 0xa1b2c3d4 (identické řazení). Druhý způsob je, že přečte hodnotu 0xd4c3b2a1 (obrácené pořadí). Pro soubor s rozlišením na nanosekundy to jsou hodnoty 0xa1b23c4d (identické řazení) a 0x4d3cb2a1 (obrácené pořadí).
- *version_major, version_minor* - určuje verzi formátu souboru (aktuální verze je 2.4)
- *thiszone* - slouží ke korekci času mezi různými časovými pásmy. Hodnota se zadává v sekundách. Pokud jsou časové údaje v GTM (UTC), je tato hodnota nastavena na 0. Pokud jsou časové údaje ve středoevropském čase (GTM + 1:00) musí být tato hodnota nastavena na -3600. V praxi se ale hodnota tohoto parametru nastavuje na 0.

- *sigfigs* - teoreticky slouží k určení přesnosti časových údajů. V praxi je tato hodnota nastavena na 0.
- *snaplen* - maximální délka „*snaplen*"pro zachycení dat. Většinou tato hodnota bývá nastavena na hodnotu 65535, ale může být i větší.
- *network* - určuje typ záhlaví na spojové vrstvě. Pro Ethernet je tato hodnota nastavena na 1.

Záhlaví zachyceného paketu - každý zachycený paket začíná tímto záhlavím. Nejedná se o samotné záhlaví zachyceného paketu, ale o popis paketu který byl zachycen:

```
typedef struct pcaprec_hdr_s {
        guint32 ts_sec;         /* timestamp seconds */
        guint32 ts_usec;        /* timestamp microseconds */
        guint32 incl_len;       /* number of octets of packet saved in file */
        guint32 orig_len;       /* actual length of packet */
} pcaprec_hdr_t;
```

Výpis 2: Syntaxe záhlaví zachyceného paketu souboru pcap

- *ts_sec* - udává datum a čas, kdy byl paket zachycen. Hodnota je ukládána v unixovém čase (čas v sekundách od 1.1.1970).
- *ts_usec* - v klasickém pcap souboru je tento parametr offset v milisekundách pro parametr ts_sec. Pokud přesáhne hodnotu 1 000 000 musí být parametr ts_sec inkrementován. V souboru, který má rozlišení na milisekundy udává tento parametr, kdy daný paket byl zachycen. Pokud přesáhne hodnotu 1 000 000 000 musí být parametr ts_sec inkrementován.
- *incl_len* - počet bajtů z daného paketu, které byly zachyceny a uloženy do souboru. Hodnota tohoto parametru by nikdy neměla být větší než hodnota parametru *orig_len* a hodnota parametru *snaplen* z globálního záhlaví.
- *orig_len* - skutečná velikost paketu při zachycení na síti. Pokud se hodnota parametru *orig_len* liší od hodnoty parametru *incl_len*. Byl paket zkrácen z důvodu překročení hodnoty parametru *snaplen* z globálního záhlaví.

Po globálním záhlaví a záhlaví zachyceného paketu už následují data samotného zachyceného paketu, poté se znovu opakuje záhlaví dalšího zachyceného paketu, které je opět následováno daty tohoto paketu. Toto řazení je zobrazeno na obrázku 11.

Některé věci v tomto formátu chybí a mohly by být užitečné: rozlišení zachycení paketu v nanosekundách, komentáře uživatele („výpadek spojení je zobrazen od paketu 1579"), informace o rozhraní (výrobce síťové karty), počet zahozených paketů (a další různé údaje ohledně počtu). Všechny tyto popsané nedostatky by měly být součástí next generation pcap (pcap-NG), který má být nástupcem formátu pcap. Tento nový formát je stále ve vývoji. Některé aplikace tento nový formát podporují již nyní. [11]

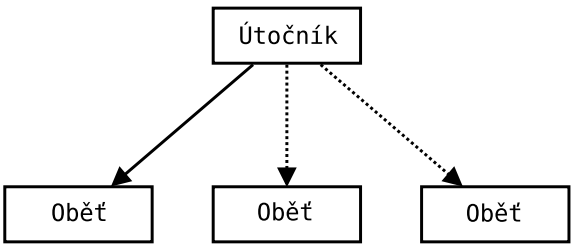
4 Aktivní útoky

4.1 Denial of Service Attack, Distributed Denial of Service Attack

Server dokáže v jednom okamžiku zpracovat pouze určité množství požadavků. Toho většinou útočník využívá a snaží se server zahltit požadavky. Cílem těchto útoků bývá většinou jedna konkretní internetová služba (email, bankovnictví, DNS, VoIP...) nebo internetové stránky. Útočník využívá chyby v implementaci TCP/IP, softwaru nebo hardwaru. [5]

Denial of Service (DoS) Attack - útok vedoucí k přerušení nebo pozastavení služeb konkrétního uživatele nebo serveru. Server není schopen odpovídat na legitimní požadavky. Existují dva základní typy útoků DoS [12, 13]:

- odeslání speciálně vytvořených paketů, které daný server neočekává a které způsobí restartování nebo zastavení systému;
- odeslání velkého množství paketů, které server nedokáže zpracovat v daném časovém úseku a dojde k vyčerpání systémových prostředků.



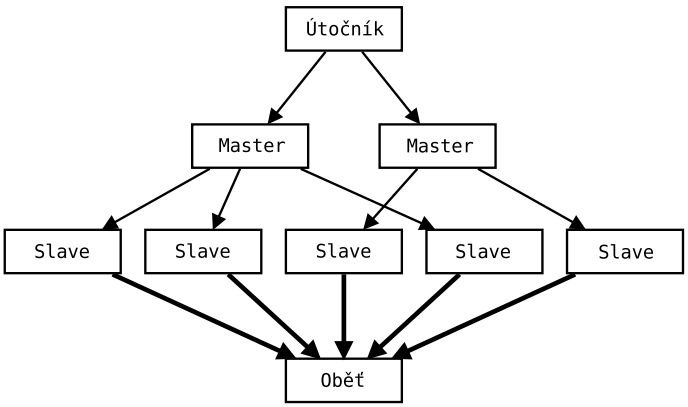
Obrázek 12: Denial of Service Attack

Distributed Denial of Service (DDoS) Attack - útočník využívá cizí počítače/servery, které má pod kontrolou (*botnet*). Ukázka *botnet* sítě je na obrázku 13. Tyto cizí počítače/servery posílají velké množství dat a požadavků (DoS Attack) cíleně na počítače/-servery dané oběti. [12, 13]

Příznaky útoku:

- neobvykle pomalá odezva sítě (při otevírání souboru nebo při přístupu na webové stránky)
- nedostupná konkretní webová stránka
- zamítnutí přístupu k libovolné webové stránce
- rapidní nárůst obdrženého spamu

Ne všechny výpadky dané služby musí znamenat útok typu DoS nebo DDoS. V dané síti může probíhat údržba nebo aktualizace systémů, může se také jednat o problémy samotné sítě. Výpadek může být také zaviněný selháním softwaru nebo hardwaru.



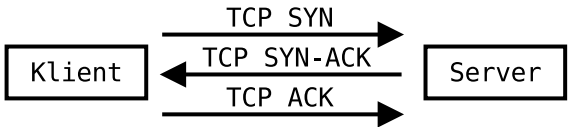
Obrázek 13: Distributed Denial of Service Attack

DoS a DDoS útoky můžeme rozdělit do pěti skupin:

1. Vyčerpání výpočetních zdrojů: šířka pásma, operační paměť, místo na disku nebo čas procesoru.
2. Změnou konfigurací: informace o směrování, změnou spanning-tree konfigurace
3. Narušením stavu informace: nevyžádané resetování TCP relací.
4. Fyzické zničení síťových komponent.
5. Zabránění komunikace mezi legitimními uživateli a obětí, tak aby nemohli dostatečně komunikovat.

4.1.1 SYN Flood Attack

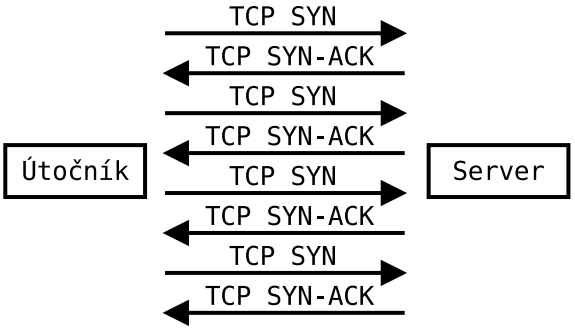
K tomuto útoku dochází při sestavování TCP spojení během *three-way handshake*. V normálním případě klient pošle požadavek (TCP SYN) pro připojení k serveru. Server rezervuje systémové prostředky a pošle paket potvrzující žádost o připojení (TCP SYN-ACK). Poté, co klient obdrží paket TCP SYN-ACK, odpoví na něj zasláním potvrzujícího paketu TCP ACK. V tuto chvíli je vytvořeno TCP spojení. Celý proces je zobrazen na obrázku 14.



Obrázek 14: Navázání TCP spojení

Během SYN Flood útoku na server pošle útočník žádost (TCP SYN) na sestavení spojení. Server rezervuje systémové prostředky a pošle paket potvrzující žádost (TCP

SYN-ACK) (viz obrázek 15). Útočník však už neodpoví paketem potvrzující spojení (TCP ACK). Těchto požadavků na spojení pošle útočník několik. Výsledek toho je, že na serveru jsou rezervované systémové prostředky pro spojení, která nikdy nevzniknou. Server tyto rezervované systémové prostředky uvolní až po vypršení platnosti. Cílem útočníka je tedy vyčerpat systémové prostředky serveru. Aby legitimní uživatele nemohli vytvořit spojení se serverem a přistoupit tak k němu.



Obrázek 15: SYN Flood útok

Existují dvě možnosti, jak způsobit nedodání potvrzujícího paketu TCP ACK serveru. První možnost už byla popsána. Útočník úmyslně nepošle potvrzující paket TCP ACK. Druhá možnost spočívá v tom, že útočník využívá falešnou IP adresu (IP Spoofing - Kapitola 4.2.1). Útočník pošle žádost o spojení (TCP SYN) s falešnou zdrojovou adresou. Server poté pošle paket TCP SYN-ACK na tuto falešnou adresu, ale odpověď se mu už z této falešné adresy nevrátí.

SYN Flood útok je obvykle v moderních síťových infrastrukturách neúspěšný. Úspěšný je pouze tehdy, pokud server rezervuje systémové prostředky pro nové spojení ihned při požadavku na navázání spojení (TCP SYN). [5, 13]

4.1.2 Ping of Death

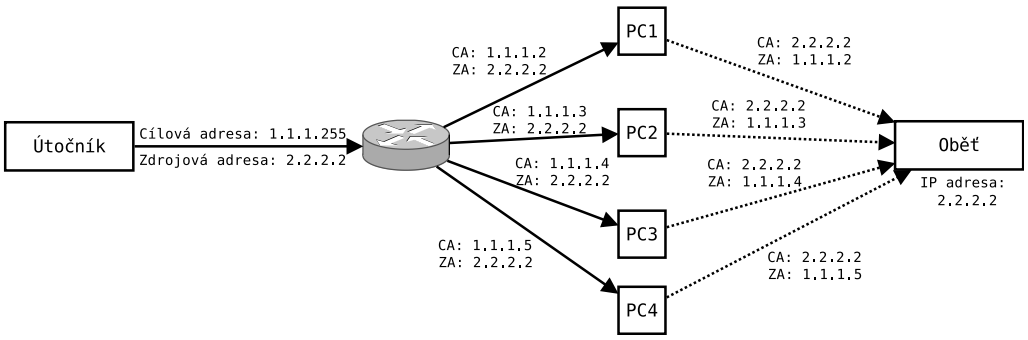
Jedná se o další typ Denial of Service útoku. Tento útok je také známý pod jménem *Long ICMP*. Útočník pošle ICMP paket, jehož velikost je větší než 64kB. Pokud takový paket obdrží zařízení oběti, dojde k zamrznutí, restartu nebo pádu operačního systému. Nové operační systémy jsou vůči tomuto útoku imunní. [13]

4.1.3 Smurf Attack

Útočník vysílá velké množství ICMP paketů „*echo-request*"na různá zařízení nebo na adresu broadcast pro různé podsítě (viz obrázek 16). Využívá k tomu falešnou zdrojovou IP adresu. Jako zdrojovou IP adresu útočník používá IP adresu oběti. Všechny zařízení ze všech podsítí odpoví na IP adresu oběti pomocí paketu ICMP „*echo-reply*". Zařízení oběti je tedy zaplaveno těmito ICMP odpovědmi a dochází k zpomalení nebo zamrznutí

celého systému. Na zařízeních od firmy Cisco je možné zabránit, aby se daná síť stala součástí těchto útoků. [13] Stačí zadat tento příkaz:

```
Router(config-if)# no ip directed-broadcast
```



Obrázek 16: Smurf útok

4.1.4 ICMP flood attack (Ping Flood)

Útok typu Ping flood spočívá v tom, že útočník zaplaví oběť ICMP „*echo-reguest*"pakety. Tyto pakety posílá s co největší frekvencí, bez toho aniž by čekal ICMP odpověď „*echo-reply*". Útok je úspěšný, pokud útočník a zařízení, která má pod kontrolou, mají celkově větší šířku pásma než oběť. Při tomto útoku dochází k zahlcení šířky pásma v obou směrech, protože zařízení oběti se bude snažit odpovídat na ICMP „*echo-reguest*"pomocí ICMP „*echo-reply*". Pokud zařízení oběti není příliš výkonné, může také dojít k velkému zatížení procesoru a tím k zpomalení celého systému. [5]

4.1.5 Back

Tento útok je mířený proti web serveru apache. Útočník pošle velké množství požadavků, které obsahují velký počet lomítek v URL adrese. Server se snaží obsloužit všechny tyto požadavky a díky tomu se může stát nedostupným nebo s omezeným přístupem pro legitimní uživatele. [13]

4.2 Spoofing Attack

Tento typ útoků spočívá v tom, že útočník se úmyslně vydává za legitimní zařízení nebo uživatele dané sítě. Cíl útočníka může být různý: získat přístup k zabezpečenému serveru, šíření škodlivého kódu, odposlouchávaní provozu, získání přístupových informací, získání citlivých dat nebo znepřístupnění některé služby. Existuje několik různých typu spoofing útoků, aby útočník dosáhl svého cíle. Mezi nejznámější patří: IP Address spoofing, ARP spoofing (ARP Poisoning), DNS spoofing a Email spoofing. Některá síťová zařízení mají implementovanou ochranu proti těmto typům útoku. Jedná se například o DHCP snooping, port security nebo Dynamic ARP inspection. [5, 14]

4.2.1 IP Address Spoofing Attack

IP address spoofing patří k nejvíce využívanému útoku typu spoofing. Útočník pošle pakety s falešnou („spoofed") zdrojovou adresou, aby tím kryl, odkud útok pochází. Tento typ útoku je často využit společně s DoS/DDoS útokem. Útok potom vypadá, že pochází z legitimní zdrojové IP adresy.

Existují dvě metody IP spoofing útoku pomocí kterých může útočník provést přetížení cíle síťovým provozem. První metoda spočívá v tom, že útočník posílá přímo provoz na oběť z mnoha různých falešných („spoofed") adres. Metoda je úspěšná, pokud je příchozí provoz větší než provoz, který dokáže oběť zpracovat. Druhá metoda pracuje s IP adresou oběti. Útočník pošle různé požadavky na mnoho různých zařízení v síti s falešnou zdrojovou IP adresou. Jako falešnou zdrojovou IP adresu útočník využívá IP adresu oběti. Všechny zařízení odpoví na tyto falešné požadavky a odešlou odpověď na IP adresu oběti. Ta nedokáže všechny tyto odpovědi zpracovat a tím dochází k přetížení systému síťovým provozem. Tato metoda je použita u DoS/DDoS útoku typu Smurf Attack.

IP address spoofing může také být použit k prolomení autentizace založené na IP adrese. Tento postup může být velice obtížný a lze ho použít pouze tam, kde je využita důvěra mezi síťovými zařazeními a vnitřním systémem. V takovém systému je použita IP adresa k ověření identity a povolení přístupu. Místo ověření identity pomocí uživatelského jména a hesla. Tímto způsobem se útočník může dostat do zabezpečeného systému a získat tím třeba nějaké citlivé informace nebo provést další typy útoků. [14]

4.2.2 ARP Spoofing Attack

ARP (Address Resolution Protocol) slouží k získání MAC (Media Access Control) adresy pro IP adresu kterou známe. V útoku typu ARP spoofing útočník pošle v místní síti falešnou zprávu ARP. Jeho cílem je propojit svou MAC adresu s IP adresou oběti v dané síti. Tím zajistí, že provoz, který je určen pro IP adresu oběti, bude poslán na jeho MAC adresu. Tento typ útoku slouží k získání informací, k úpravě nebo zastavení provozu v místní síti. ARP spoofing slouží, také k usnadnění jiným typům útoku: DoS/DDoS, neoprávněnému přístup a útoky typu man-in-the-middle. ARP spoofing lze provést pouze v LAN (Local Area Networks) sítích, kde je použit Address Resolution Protocol. [14]

4.2.3 DNS Spoofing Attack

DNS (Domain Name System) je systém, který propojuje doménové jména s IP adresami. Zařízení připojená k internetu nebo k privátní síti spoléhají na DNS při překladu URL, e-mailové adresy a dalších doménových jmen na odpovídající IP adresu. Útočník pomocí jiných typů útoků změní IP adresu DNS serveru na zařízení oběti. Pokud oběť bude chtít přistoupit například na stránky svého internetového bankovnictví, pošle dotaz na tento falešný DNS server. Tento falešný DNS server odpoví oběti a pošle ji falešnou IP adresu web serveru. Na falešném web serveru může být vytvořena úvodní přihlašovací stránka, která vypadá identicky jako ta na skutečném webu s internetovým bankovnictvím. Po-

kud oběť vyplní své přihlašovací údaje na tomto falešném serveru, dostávají se do rukou útočníka. Nemusí se jednat pouze o internetové bankovnictví, ale třeba o platbu kartou, přihlášení k emailu, přihlášení k sociální síti a mnoho dalšího. Falešný web server může být využit ještě druhým způsobem. Útočník na tento falešný web umístí různé viry, červy a další škodlivý software, pomocí kterého dojde k napadení systému oběti. [14]

4.2.4 Email Spoofing Attack

Tento typ útoku bývá využíván společně s Phishing útokem nebo při zasílání nevyžádané pošty (spam). Útočník změní nebo podvrhne jméno odesílatele. Email poté vypadá jako od legitimní osoby nebo společnosti.

Phishing útok - tento typ útoků spadá pod sociální inženýrství. Využívá se k zasílání podvodných emailů, které působí velmi důvěryhodně a snaží se oběť oklamat svou realističností. Většinou email obsahuje odkaz na falešný web a žádá vás, aby jste vyplnili své osobní nebo přihlašovací údaje z důvodu: zvýšení zabezpečení, vypršení platnosti hesla, váš účet se stal terčem útoku, apod. Útočník díky vaší neopatrnosti může lehce získat vaše údaje. [5]

4.3 Malware

Slovem malware se označuje veškerý škodlivý software. Tento název vznikl z anglických slov *malicious* a *software*. Do této kategorie například patří: viry, červi, trojské koně, spyware, adware, back door apod. Jedná se o škodlivý kód nebo software, který vznikl speciálně za účelem poškození, narušení, krádeže nebo pro další nelegitimní úkony. Cílem malware jsou data, hostitelské zařízení, sítě a síťové zařízení.

Existuje několik různých způsobů, jak může dojít k infikování systému. Malware může infikovat systém tím, že je v balíku s dalšími programy nebo připojený jako makro k nějakému souboru. Další druh malware může být nainstalován využitím známých zranitelností v operačním systému, síťovém zařízení nebo jiném softwaru (díry ve webovém prohlížeči). V takovém případě stačí pouze, aby uživatel navštívil infikované webové stránky. Naprostá většina malware je nainstalována do zařízení za pomocí nějaké akce od uživatele. Zde můžeme zařadit například otevření přílohy v emailu nebo stahování souborů z internetu.

Malware nemůže poškodit fyzický hardware systému nebo síťových zařízení, ale může dojít k poškození dat a softwaru umístěného na zařízení. Malware by neměl být zaměňován s vadným softwarem, který je určený pro legitimní účely, ale obsahuje vady nebo chyby. [15, 16]

4.3.1 Viruses

Počítačový vir je druh malware, který se šíří tím, že vloží kopii sebe sama do jiného programu a stane se tak jeho součástí. Při přenosu takového programu, dochází k napadení dalších počítačů. Viry se tedy šíří prostřednictvím sítě, diskových úložišť, sdílení souboru

nebo pomocí emailů. Viry mají různý rozsah škod, které mohou způsobit. Od poškození souboru nebo softwaru až po DoS útok.

Téměř všechny viry jsou připojeny k spustitelným souborům, což znamená, že virus může být v systému přítomen, ale není aktivní a není schopen se dál šířit dokud uživatel neotevře nebo nespustí infikovaný soubor nebo program. Při spuštění infikovaného programu, program proběhne normálně, ale spustí se i škodlivá část kódu. Za normálních okolnostní tedy soubor nebo program dále fungují, poté co jsou napadeny virem. Nicméně některé viry přepíší program uživatele kopií sama sebe a tím tento program zničí. [15, 16]

4.3.2 Worms

Počítačové červi jsou podobné jako viry v tom, že replikují kopii sama sebe a mohou způsobit stejný rozsah škod. Na rozdíl od virů, které potřebují k šíření hostitelský soubor nebo program, červi jsou samostatný software a dokáží se propagovat sami. Červi vstoupí do systému prostřednictvím zranitelnosti a chyb tohoto systému. K šíření využívají soubory nebo jiné dopravní prvky v systému, to jim umožňuje cestovat bez jakékoliv pomoci. [15, 16]

4.3.3 Trojans

Jedná se o škodlivý software, který se tváří jako legitimní. Uživatelé si obvyklé myslí, že se program načítá nebo spouští, ale mezitím se do počítače instaluje škodlivý software. Poté, co je tedy tento škodlivý software aktivován, může být proveden libovolný počet útoků na hostitelský systém. Na uživatele například vyskakuje plno oken za účelem odvedení pozornosti. Uživatel se je snaží zavřít, ale mezitím dochází k mazání souboru, krádeže dat nebo aktivaci a šíření dalšího škodlivého softwaru. Trojské koně jsou také známé, že vytváří zadní dvířka (backdoor) pro útočníka. [16]

4.3.4 Backdoor

Backdoor slouží k vytvoření přístupu do systému a k vyhnutí se autentizačním mechanismům. Některé backdoor jsou vytvořeny samotnými programátory při vývoji softwaru, některé se do softwaru dostanou po překonání zabezpečení systému, pomocí škodlivého softwaru (viry, červi, trojské koně). Tímto způsobem si útočník vytvoří v systému díru pro snadnější a pozdější nepřetržitý přístup k tomuto hostitelskému systému. [16]

4.3.5 Spyware

Jedná se o software, který byl vyvinut s cílem získávání citlivých informací ze zařízení obětí bez jejich vědomí. Tyto získané citlivé informace potom zasílá útočníkovi. Jde především o čísla bankovních účtů a karet, které může útočník zneužít. Spyware se do zařízení dostane většinou při instalaci nějakého programu nebo při kliknutí na některá vyskakovací okna na nedůvěryhodné internetové stránce. [15]

4.3.6 Rootkit

Rootkit je sada nástrojů sloužící pro zakrytí škodlivého softwaru a k povolení administrátorského přístupu na hostitelském zařízení. Může být součástí výše popsaných typů malware nebo sloužit k vytvoření prostředí pro jejich zavedení do systému. Je velmi těžké detekovat tento typ malware. Obvykle je aktivován ještě před úplným spuštěním operačního systému. Rootkit umožňuje útočníkovi zavádět do systému skryté soubory, procesy, účty nebo instalovat další škodlivý software. Je schopen také zachytit data z terminálu nebo klávesnice. [15]

4.4 Google Hacking

Google hacking je velmi zajímavá technika. Využívá se k tomu fulltextový vyhledávač od společnosti Google, pomocí něhož útočník hledá bezpečnostní slabiny na internetu. Ve většině případů se jedná o dva typy bezpečnostních slabin, které můžeme na webu najít: slabina v softwaru nebo špatná konfigurace. Google Hacking Database (GHDB) je databáze dotazů, které identifikují citlivá data. Útočník pomocí pokročilých google operátorů vytváří sofistikovaný dotaz a snaží se vyhledat a využít známé slabiny v systému. Tyto operátory můžeme různě kombinovat:

- intitle, allintitle, inurl, allinurl, filetype, allintext, site, link, inanchor, numrange, daterange, author, group, insubject, msgid.

Syntaxe těchto operátorů a ukázka přístupu na tiskárnu od firmy HP vypadá takto:

```
operator:search_term
```

```
inurl:hp/device/this.LCDispatcher
```

Útočník se takto může připojit například k internetové(bezpečnostní) kameře a sledovat co se zrovna děje v daném objektu. Při použití nástroje *whois* může také zjistit, kde daná IP adresa sídlí. Útočník má tedy přehled o objektu a zná i jeho adresu. Pomocí jiného dotazu se útočník může dostat do domácího směrovače a změnit nastavení, což může být dále využito k jiným typům útoku. Mnoho těchto zařízení má nastavené tovární přihlašovací údaje a proto není složité se dostat přes autentizaci. Další zařízení, ke kterému se takto může útočník připojit, je tiskárna.

Tiskárny a domácí směrovače nejsou jediná zařízení, ke kterým se dá takto lehce připojit. Mnohem nebezpečnější je pokud se útočník připojí takto k nějakému zařízení typu: Firewall, IDS (Intrusion Detection System) nebo UPS (Uninterruptible Power Supply/-Source) - zdroj nepřerušovaného napájení. Na těchto zařízeních už útočník může způsobit velké škody a provést útoky s mnohem většími dopady na danou síť, na rozdíl od přístupu k tiskárně. Dalším cílem mohou být pobočkové ústředny, které může překonfigurovat, vypnout nebo zjistit citlivá data uživatelů.

Mnoho citlivých informací je také uloženo v chybových hlášeních nebo lozích, ke kterým se lze takto dostat. S tímto je spojena i další možnost využití pokročilých operátorů

a to s SQL dotazy. Útočník takto může zjistit strukturu databáze a pokusit se ji vymazat nebo může zjistit heslo pro připojení k této databázi.

Útok typu Google hacking lze provést v dnešní době prakticky odkudkoliv, protože většina mobilních zařízení má přístup k internetu. K provedení takovéhoto útoku stačí tedy pouze zařízení s internetovým prohlížečem a přístupem k internetu. [17, 18]

4.5 Útoky na IPv6

Tato podkapitola je zaměřena čistě jen na útoky prováděné na IPv6 protokolu. Předchozí popis útoků byl ve většině případů společný pro IPv4 i IPv6. Ačkoliv protokol IPv6 je už poměrně starý, jeho nasazování a využití není ale zatím příliš obrovské. S jeho postupným zavaděním se objevují i nové problémy, které je potřeba řešit. Zejména v zabezpečení síťové infrastruktury.

4.5.1 Zneužití zpráv Router Advertisement (RA)

Zneužití zpráv *Router Advertisement* je jeden z nejznámějších útoků v prostředí IPv6 sítích. Zařízení v IPv6 sítí pomocí zprávy *Router Solicitation* zažádá o přidělení údajů potřebných pro komunikaci v rámci dané sítě. Tyto údaje jsou rozeslány všem zařízením v síti směrovačem pomocí zprávy *Router Advertisement*. Útok může být také zaměřený jen na konkrétní unicastové adresy.

Zneužití těchto zpráv spočívá v tom, že za legitimní směrovač se může vydávat jakékoliv zařízení v dané síti a pomocí upravené zprávy *Router Advertisement* může útočník propagovat falešné konfigurační informace. Takto může útočník přesměrovat provoz na sebe, odepřít konektivitu do IPv6 světa nebo odkázat oběť na falešný server, který se bude tvářit jako legitimní server.

Zařízení podporující IPv6 může mít na jednom rozhraní nakonfigurovaných několik IPv6 adres. Tuto vlastnost lze využít k dalšímu zneužití zpráv *Router Advertisement*. Útočník může totiž periodicky posílat *Router Advertisement* s různými prefixy a operační systém musí tyto zprávy zpracovat. To má za následek nakonfigurování další IPv6 adresy na rozhraní. Při dostatečně rychlém generování těchto zpráv dojde k velkému vytížení procesoru oběti, což může vést až k restartování systému. Tento typ útoku je známý pod názvem **RA Flood**. Zprávu *Router Advertisement* lze rozšířit o volbu *Route Information Option*, která má v sobě uložené podrobnější informace i směrování. Takto lze vložit do směrovací tabulky několik (potencionálně až stovky) záznamů pomocí jednoho paketu. [19]

4.5.2 Zneužití Extension Headers

Extension Headers slouží pro rozšiřování protokolu IPv6 o další zajímavé vlastnosti. Kromě základní hlavičky, lze paket IPv6 tedy ještě doplnit o další rozšiřující hlavičky. Většina ISP (Internet Service Provider) pakety s *Extension Headers* rovnou zahazují, aby předešly problémům, které mohou vzniknout.

Extension Headers lze zneužít několika způsoby. Jedním z nich je jejich cílené vkládaní do těla paketu. Útočník takto doručí oběti data, která by za normálních okolnosti neprošla přes filtrační mechanismus. Využívá k tomu vkládání za sebe několika *Extension Headers*. Tím docílí toho, že filtrační mechanismus nedokáže rozpoznat jaký typ protokolu, je přenášen v paketu IPv6.

Dalším ze způsobů, jak zneužit *Extension Headers*, je pomocí hlavičky fragmentace. Cílem je paket účelově rozdělit do fragmentů. Takovýto paket poté projde přes většinu jednoduchých paketových filtrů, protože tyto paketové filtry neumí provádět rekonstrukci fragmentů.

Útočník dále může využít zprávu *Packet Too Big*, která je zaslána zařízením, pokud je přijatý paket příliš velký a dané zařízení ho nedokáže zpracovat. Protokol IPv6 vyžaduje, aby koncové zařízení uměly zpracovat paket, jehož minimální délka je nastavena na 1280 bajtů. Útočník pošle zprávu *Packet Too Big*, serveru s kterým chce oběť komunikovat. V této zprávě informuje server, že cesta k oběti má menší MTU než 1280 bajtů. Pokud se oběť bude chtít připojit na tento server, server mu bude zasílat odpovědi s vloženou hlavičkou fragmentace. Vzhledem k tomu že většina ISP zahazuje pakety s *Extension Headers*, je velká pravděpodobnost, že útočník zamezí komunikaci mezi serverem a obětí. Pokud pakety s *Extension Headers* dorazí až k oběti, tak nastává druhý problém. Internetové prohlížeče většinou odpověď serveru s hlavičkou fragmentace neberou, jako odpověď na svůj požadavek o navázání spojení.

Přepínače jsou navrženy tak, aby většinu provozu zpracovávaly pomocí hardwaru. Nicméně pokud na přepínač dorazí paket, který hardware neumí zpracovat, je tento paket poslán na procesor ke zpracování. IPv6 paket, který vždy zpracovává procesor je například paket s *Extension Headers Hop-by-Hop*. Ostatní pakety s *Extension Headers* většinou přepínač přepošle pomocí hardware. Problém nastává ve chvíli, kdy od přepínače požadujeme, aby prováděl kontrolu a validaci těchto paketů. Takovéto pakety poté musí zpracovat procesor. Tímto způsobem může dojít k velkému vytížení procesoru. [20]

4.5.3 Zneužití protokolu MLD (Multicast Listener Discovery)

Protokol MLD je využíván v lokálních sítích pro přihlášení do multicastových skupin. Používají se k tomu tři zprávy: *Query, Report, Done*. Vzhledem k jeho využití nelze tento protokol příliš filtrovat a samotný neobsahuje žádné metody pro zabezpečení. Tyto vlastnosti mohou být zneužity k provedení útoku.

Útočník může prostřednictvím protokolu MLD zmapovat danou síť. Nebude skenovat adresu po adrese, jako u IPv4 adres, ale využije k tomu zprávu MLD *Query*. Pokud útočník vyšle tuto zprávu v dané lokální síti, všechna zařízení odpoví na tuto zprávu. Pošlou své informace o tom, do kterých multicastových skupin jsou přihlášené. Tímto způsobem útočník zjistí link-local adresy daných zařízení. Ty poté může využít k útoku typu *Port Scanning*, protože zařízení implicitně naslouchají na všech IPv6 adresách.

Další ze způsobů, jak zneužít protokol MLD, je zamezení příjmů provozu typu multicast. Útočník se prohlásí za falešný *Querier* (směrovač, který má na starost přeposlání provozu typu multicast). *Querier* se stává směrovač s nejnižší IP adresou. Většina směro-

vačů si vytváří IPv6 adresu s použitím EUI-64 algoritmu a linkové adresy, proto není pro útočníka problém si nakonfigurovat nižší IPv6 adresu z daného rozsahu.

Jako většina protokolů se i tento dá zneužít k přetížení síťových zařízení. Pro efektivní šíření provozu v lokální síti je potřeba mít zapnutý MLD *Snooping*. Při použití MLD *Snooping* musí síťové zařízení všechny zprávy MLD *Report* a *Done* posílat ke zpracování procesoru. Pokud útočník dokáže generovat zprávy *Query* dostatečně rychle, může způsobit značné potíže danému zařízení, při zpracovávání odpovědi na tuto zprávu. Přibližně 3000 těchto zpráv za sekundu dokáže zcela vytížit zařízení Cisco Catalyst 3650. [21]

4.5.4 Útoky na Neighbor Cache

Tento typ útoku se zaměřuje na přeplnění *Neighbor Cache* a k vytížení daného zařízení. Útok může být proveden lokálně nebo vzdáleně přes internet. *Neighbor Cache* využívá každé zařízení, které pracuje s IPv6. Hlavní položky této struktury jsou IPv6 adresa a k tomu odpovídající MAC adresa, dále je ukládán i stav (*Reachable, Stale, Delay, Probe*) ve kterém se adresa zrovna nachází.

Lokální útok je založený na generování falešných zpráv *Neighbor Solicitation* a *Neighbor Advertisement*. Síťové zařízení si pro každou falešnou IPv6 adresu vytvoří záznam v *Neighbor Cache*. Tímto způsobem může dojít k vyčerpání místa v *Neighbor Cache*, které je určené pro legitimní záznamy. Legitimnímu zařízení může být takto odepřena IPv6 konektivita. Některé přepínače po naplnění *Neighbor Cache* nové záznamy rovnou zahazují, některé naopak pošlou tyto pakety procesoru ke zpracování. Takto dojde i k velkému vytížení daného přepínače.

Nejjednodušší forma je generování zpráv *Neighbor Solicitation* s falešnou adresou. Zařízení, které dostane tuto zprávu si falešnou adresu uloží do *Neighbor Cache* ve stavu *Delay* a vyčká na ověření dostupnosti této adresy. Operační systém provede toto ověření během pár sekund a falešná adresa je odstraněna z *Neighbor Cache*. Tento typ útoku dokáže zařízení zatížit, ale nevyčerpá zdroje určené pro *Neighbor Cache*. Mnohem sofistikovanější způsob je v případě, že falešná adresa je skutečně dostupná. Tato falešná adresa je poté v *Neighbor Cache* ve stavu *Reachable*. Byla tedy ověřená její dostupnost a bude uchovaná po výrazně delší dobu (řádově hodiny), než tomu bylo v předchozí variantě. Útok lze poměrně lehce provést na většině systémech Linux. Jelikož na rozhraní může být nakonfigurováno několik IPv6 adres, je snadné takto vytvořit falešné adresy. Poté už jenom stačí využít příkaz *ping6*, kterým docílíme, že daná falešná adresa bude ověřena.

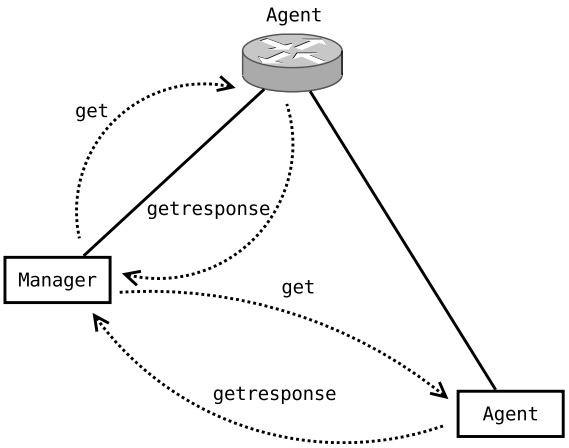
Vnější útok je založen na posílání paketů na různé adresy do koncové sítě. Hraniční směrovač poté musí uložit tyto adresy do *Neighbor Cache* a zaslat zprávy *Neighbor Solicitation* za účelem ověření dostupnosti těchto adres. Vnější útok má prakticky stejný výsledek jako prvně zmiňovaný lokální útok. Nedojde tedy k vyčerpání zdrojů určených pro *Neighbor Cache*, ale dojde k docela velkému vytížení tohoto zařízení. Cílem tohoto útoku nemusí být pouze servery nebo koncoví uživatelé. Tímto útokem může být narušena i samotná infrastruktura dané sítě. [22]

5 Sběr síťových a systémových informací

Pro sběr síťových a systémových informací jsem z velké části použil protokol SNMP (Simple Network Managment Protocol) a dále pak informace získané z operačního systému Linux. Protokol SNMP jsem zvolil z důvodu, že tento protokol je podporován mnoha zařízeními a taky kvůli hodně informacím, které se dají zjistit tímto protokolem o daném zařízení.

5.1 SNMP (Simple Network Managment Protocol)

Protokol SNMP byl navržen pro správu a řízení síťových zařízení (směrovače, přepínače, servery), lze ale pomocí něho spravovat i mnoho dalších zařízení, které mají podporu SNMP. Protokol vznikl v roce 1988. Pracuje na modelu klient/server. Klient je označován jako SNMP manager. Server je označován jako SNMP agent. SNMP manager zasílá požadavky na SNMP agenta, který mu pošle dané informace zpět (viz obrázek 17). Protokol SNMP běží standardně na UDP portu 161 (SNMP agent) a na UDP portu 162 (SNMP trap), může ale běžet i na stejných portech na TCP. Secure SNMP je verze, která využívá zabezpečení (SNMPv3) a běží na TCP/UDP portech 10161 (SNMP agent) a na TCP/UDP portech 10162 (SNMP trap). SNMP protokol může být využit i lokálně. SNMP manager i agent jsou na jednom zařízení zároveň. [23]



Obrázek 17: SNMP - komunikace mezi managerem a agentem

5.1.1 Verze SNMP

- **SNMP verze 1 (SNMPv1)** - jedná se o první verzi tohoto protokolu. Hlavní nevýhoda této verze spočívá v zabezpečení. Pro zabezpečení využívá heslo (public), které je posíláno v podobě čistého textu. Heslo lze tedy jednoduše zjistit analýzou zachyceného paketu. SNMPv1 používá tři komunity: *read-only*, *read-write* a *trap*.

- **SNMP verze 2 (SNMPv2)** - tato verze je označována jako **SNMPv2c** (*community-string-based*). Tato verze SNMP protokolu je nejvyužívanější i přesto, že heslo opět zasílá jen v podobě čistého textu. V této verzi byla implementována i kontrola doručení. Pro autentizaci využívá *community-string*.
- **SNMP verze 3 (SNMPv3)** - zatím nejnovější verze protokolu SNMP. Její hlavní výhodou je zabezpečení. Pro autentizaci využívá jméno, heslo a šifrování (AES).

5.1.2 Databáze MIB (Management Information Base)

Všechny informace, s kterými SNMP protokol pracuje, jsou uloženy v MIB databázi. Jedná se objektově orientovanou stromovou strukturu, ve které jsou uloženy informace o daném zařízení s jednoznačnou identifikací. K tomu slouží identifikátory objektů OID (object identifier). Pro správnou komunikaci musí být MIB databáze totožná u SNMP agenta i managera. Základní objekty jsou uloženy ve větvi *mgmt* (managment). Každý výrobce si může definovat vlastní objekty v této databázi. Ty se pak nacházejí ve větvi *private*. [23]

5.1.3 SNMP zprávy

SNMP protokol pracuje s několika typem zpráv. Některé jsou dostupné pouze pro SNMPv2 a SNMPv3. Tyto zprávy jsou přenášeny pomocí PDU (Protocol Data Unit). Jedná se o formát zpráv, který slouží k posílání a přijímání informací mezi SNMP agentem a managerem. [23]

- **get** - jedná se o základní zprávu, kterou posílá manager -> agentovi. Manager zasílá požadavek na jednu konkretní informaci. Agent poté odpoví pomocí zprávy **getresponse**. Využívá se k tomu příkaz *snmpget*.
- **set** - pomocí této zprávy můžeme měnit nebo nastavit hodnoty dané proměnné. Lze takto také přidat řádek do tabulky. Manager může takto nastavit třeba název zařízení (agenta). Využívá se k tomu příkaz *snmpset*.
- **getnext** - tato zpráva slouží pro získání více informací najednou. Pro každý objekt z MIB se použije samostatná zpráva **get** a **getresponse**. Nemusí se tedy posílat několik zpráv **get** samostatně. Definuje od kterého kořene má SNMP vracet informace. Využívá se k tomu příkaz *snmpwalk*.
- **getbulk** (SNMPv2 a SNMPv3) - jedná se o vylepšenou zprávu **getnext**. Agent v některých případech nedokáže zaslat celou odpověď se všemi informacemi a vrátí zprávu s chybovým hlášením. Pomocí zprávy **getbulk** dokáže SNMP rozdělit odpověď na několik částí. Využívá se k tomu příkaz *snmpbulkget*.
- **notification** (SNMPv2 a SNMPv3) - tato zpráva má opačný směr komunikace, než je tomu u zpráv **get** a **set**. Zde agent zasílá informace manageru. SNMP využívá dva typy těchto zpráv: **trap** a **inform**.

- **trap** - pomocí této zprávy informuje agent managera, že došlo k významné události - překročení některé hodnoty po předchozím nastavení limitu. Je nutné mít nakonfigurovanou IP adresu, kam se má zpráva **trap** zaslat. Většinou se jedná o IP adresu managera. Využívá se k tomu příkaz *snmptrap*.
- **inform** (SNMPv2 a SNMPv3) - tato zpráva se využívá při použití více manageru v jedné síti. Slouží k výměně informací uložených v jejich MIB databázi.
- **report** (SNMPv2 a SNMPv3) - pomocí této zprávy je zajištěna komunikace mezi agenty, pokud nastane problém při zpracování zpráv.

5.1.4 Prohlížeč MIB databází

MIB databáze obsahují velké množství informací. Pro přehlednější a rychlejší vyhledávání byl použit prohlížeč MIB od firmy iReasonic [24]. Využívá grafické prostředí a je dostupný pro většinu operačních systémů. Jediná nutnost je mít nainstalované prostředí Java. Podporuje všechny verze SNMP (SNMPv1, SNMPv2c a SNMPv3). Kromě základní databáze MIB, umožňuje načíst i MIB databáze jednotlivých výrobců. Dále pak byl použit online SNMP Object Navigator [25] na stránkách společnosti Cisco. Tento nástroj byl zvolen zejména pro identifikaci daných OID a také pro procházení stromové struktury MIB databáze za účelem nalezení vhodných parametrů pro monitorování a vyhledávání anomálií.

5.2 Informace získané z OS Linux

Operační systém Linux nabízí mnoho možností, jak získat informace o daném zařízení. Není nutné instalovat další software s grafickým rozhraním. Vše je možné získat přímo z terminálu. To v sobě ukrývá další výhodu v podobě zpracování těchto informací. Informace mohou být zpracovány skripty napsané v jazycích *Bash* nebo *Perl*, ale i pomocí mnoha dalších. Operační systém Linux ukládá spoustu informací do souborů v čitelné podobě. Stačí tedy daný řetězec jen zpracovat a vyfiltrovat potřebné informace. Velmi mnoho síťových informací lze nalézt v těchto dvou adresářích:

```
/proc/net/
/sys/class/net/eth0/statistics/
```

Nalezneme zde například tyto informace: počet přijatých a odeslaných bajtů, počet přijatých a odeslaných paketů, počet zahozených paketů, informace ohledně chyb, navázaná TCP spojení pro IPv4 i pro IPv6, informace o ICMP, tabulka ARP, informace pro směrování, informace o Wi-Fi připojení a mnoho dalšího.

Další možností jak získat potřebné informace, je pomocí různých příkazů a výpisů do terminálu, které je možné opět zpracovat skripty. Mezi tyto příkazy patří: **ps -ax** (počet spuštěných procesů), **free** (využití paměti RAM), **df** (využití místa na disku a na dalších médiích), **top** (vytížení procesoru). Existuje mnoho dalších balíků, které lze doinstalovat a získat tak další příkazy pro vypisování informací.

6 Vyhledávání anomálií v získaných datech

Většina IDS/IPS systémů pracuje na porovnávání signatur. Využívá tedy pravidel na základě kterých určí, zda se jedná o povolený nebo nepovolený provoz. Některé systémy umožňují detekci anomálií pomocí preprocesoru. Preprocesor je naprogramovaný modul pro daný IDS/IPS systém, který dokáže detekovat anomálie. Tyto preprocesory nejsou příliš dostupné. Jeden takovýto preprocesor pro systém Snort naprogramovali polští programátoři. Jedná se o projekt Snort.AD [26]. V současnosti je tento projekt pozastaven. Tato diplomová práce se nezaměřuje na tyto známé systémy a preprocesory, ale na detekci anomálií pomocí grafů. Pro vyhledávání anomálií byl zvolen nástroj RRDtool. Tento nástroj má v sobě implementovanou metodu *Holt-Winters*, který slouží pro detekci anomálií v reálném čase.

6.1 Metoda Holt-Winters

Metoda byla navržena Charlesem C. Holtem a následně vylepšena Peterem R. Wintersem. Metoda Holt-Winters slouží k predikci vývoje časové řady. Pracuje na principu trojitého exponenciálního vyhlazování. Jedná se tedy o matematicky model pomocí něhož se dá určit, zda data vyhovují danému modelu či nikoliv. Model vytváří predikci vývoje dat. Na základě predikce vytváří interval spolehlivosti. Pokud data nesplňují kritéria modelu a jsou tedy mimo interval spolehlivosti, algoritmus tyto data označí za nevhodné (anomálie). [6]

Predikce (1) je suma tří koeficientů: základ (2), lineární trend (3), sezonní trend (4). Určení povolené odchylky časové řady (5). Určení intervalu spolehlivosti (6).

$$\hat{y}_{t+1} = a_t + b_t + c_{t+1-m} \tag{1}$$

$$a_t = \alpha(y_t - c_{t-m}) + (1-\alpha)(\alpha_{t-1} + b_{t-1}) \tag{2}$$

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \tag{3}$$

$$c_t = \gamma(y_t - a_t) + (1-\gamma)c_{t-m} \tag{4}$$

$$d_t = \gamma|y_t - \hat{y}_t| + (1-\gamma)d_{t-m} \tag{5}$$

$$(\hat{y}_t - \delta_- \cdot d_{t-m}, \hat{y}_t + \delta_+ \cdot d_{t-m}) \tag{6}$$

$y_1, y_2..., y_t$ - hodnoty časové řady
α, β, γ - parametry adaptace
m - perioda sezonního trendu (m=288 pro jeden den, s krokem 5 minut)
d_t - předpovězená odchylka v čase t
δ_-, δ_+ - určují toleranci získaných hodnot k hodnotam předpovídaným

6.2 Nástroj RRDtool

Tento nástroj je ve většině případů používán pouze pro ukládání a vykreslování monitorovaných dat. Je součásti několika velkých monitorovacích systému jako je: Nagios, Cacti, Munin nebo NTop. Jedná se OpenSource nástroj šířený pod licencí GNU General Public License. Hlavním vývojářem je Tobias Oetiker. Nástroj je napsán v jazyce C. Poslední aktuální verze RRDtool je 1.5.0-rc2 z února 2015.

RRDtool je velmi univerzální nástroj pro monitorování veličin v reálném čase. Lze pomocí něho monitorovat prakticky cokoliv z čeho lze získat číselnou hodnotu. RRDtool ukládá data do RRDs (Round-Robin Database), která pracuje na principu kruhového bufferu. Výhoda této databáze spočívá v konstantní velikosti. Její velikost je určena při vytvoření a nezvětšuje se v čase s nově příchozími daty. Nevýhoda této databáze je v uchovávání starých hodnot. Po naplnění databáze jsou nejstarší hodnoty přepsány novými. Z tohoto důvodu je možné databázi exportovat do XML souboru a vytvořit si takto zálohu. RRDtool nemá žádné grafické prostředí pro práci s ním. Práce s tímto nástrojem je založena na skriptech. Na výběr je docela velké množství jazyků, ve kterých lze pracovat s RRDtool: Bash, Perl, Python, Ruby, Lua a existuje podpora i pro Javu. Práci s tímto nástrojem lze rozdělit do tří kroků. V prvním kroku se nadefinuje a vytvoří databáze. V druhém kroku se naplní databáze daty a v posledním kroku se vykreslí hodnoty uložené v databázi do grafu. [27]

6.3 Vytvoření databáze (funkce CREATE)

Pro vytvoření RRDs databáze slouží funkce CREATE. Syntaxe je následující:

```
rrdtool create filename [--start|-b start time] [--step|-s step] [--no-overwrite]
DS:ds-name:GAUGE | COUNTER | DERIVE | ABSOLUTE:heartbeat:min:max
RRA:CF:xff:steps:rows
```

Výpis 3: Syntaxe funkce CREATE

Základní parametry jsou: *filename*, *start*, *step* a *no-ovewrite*. *Filename* určuje název databáze. Měl by končit příponou rrd. *Start* udává čas od kdy se začnou ukládat první data. Hodnota je v sekundách od 1.1.1970 nebo lze použít N, které slouží pro okamžitý start databáze po vytvoření. *Step* určuje dobu, po které databáze bude očekávat další data. Hodnota je v sekundách. *No-overwrite* zajistí, aby nedošlo k přepsání databáze se stejným jménem.

DS*(Data Source)* slouží pro ukládání více veličin. *Ds-name* je název proměnné a délka může být maximálně 19 znaků $(a-z, A-Z, 0-9, \_)$. Následuje DST*(Data Source Type)*, který určuje jaký bude zdroj dat. Existují 4 DTS: *GAUGE, COUNTER, DERIVE* a *ABSOLUTE*. *GAUGE* slouží pro základní ukládání hodnot (teplota, vytížení CPU). Tu hodnotu, kterou dostane, uloží do databáze. *COUNTER* se využívá pro hodnoty, které neustále rostou (čítače). Obsahuje ověření přetečení pro 32-bitové a 64-bitové čítače. Výsledná hodnota je poté udávána jako hodnota za sekundu. *DERIVE* je podobné *COUNTER*, ale může nabývat i záporných hodnot a neobsahuje kontrolu přetečení čítače. *ABSOLOUTE* je využíváno u čítačů, které jsou po každém přetečení resetovány. Další parametr je *heartbeat*,

který udává jak dlouho bude databáze čekat na novou hodnotu. Pokud nová hodnota nedorazí do hodnoty *heartbeat*, je tato hodnota označena jako neznáma a do databáze se zapíše NAN (Not A Number). Slouží k tomu, aby se do databáze nezapisovaly nesmyslné hodnoty, pokud je monitorované zařízení nedostupné. Pomocí *min* a *max* nastavíme v jakém rozsahu mohou být ukládány hodnoty do databáze.

RRA(Round-Robin Archive) určuje velikost databáze. Konsolidační funkce (*CF*) může nabývat 4 hodnot: *AVERAGE, MAX, MIN, LAST*. Pokud je výsledná hodnota složena z více hodnot, konsolidační funkce určí, jak se výsledná hodnota uloží do databáze. Jestli jako průměr hodnot, maximální, minimální hodnota nebo jestli se uloží jen poslední hodnota ze získaných hodnot. Parametr *xff* určuje kolik hodnot může být neznámých, aby bylo možné vytvořit výslednou hodnotu. Parametr *xff* je v rozmezí od 0 do 1. Parametr *steps* nastavuje z kolika hodnot se bude skládat výsledná hodnota. Velikost databáze závisí na parametru *rows*. Tento parametr určí kolik bude mít databáze řádků. [27] Ukázka vytvoření databáze:

```
rrdtool create vytizeniCPU.rrd --start N --step 60 --no-overwrite
DS:cpu:GAUGE:120:0:100
RRA:AVERAGE:0.5:1:1440
```

Výpis 4: Ukázka vytvoření databáze

6.4 Naplnění databáze (funkce UPDATE)

Pomocí funkce UPDATE naplníme databázi. Syntaxe je následující:

```
rrdtool update filename
[--template|-t ds-name[:ds-name]...]
N|timestamp:value[:value ...]
```

Výpis 5: Syntaxe funkce UPDATE

Parametr *filename* slouží pro zvolení databáze, do které se bude zapisovat, protože může existovat více databází. Pomocí *template* lze změnit pořadí, ve kterém se budou ukládat DS. Poslední parametr udává čas, kdy bude hodnota uložena. Opět lze použít N pro aktuální čas. Za uvedeným časem následuje daná hodnota, která se bude ukládat do databáze. [27]

Ukázka naplnění databáze:

```
rrdtool update vytizeniCPU.rrd
N:$hodnota
#v promenne hodnota je ulozeno vytizeni CPU
```

Výpis 6: Ukázka naplnění databáze

6.5 Vykreslení grafu (funkce GRAPH)

Poslední funkce, která byla použita, je funkce GRAPH. Tato funkce slouží pro vykreslení hodnot, které jsou uložené v databázi. Funkce GRAPH obsahuje mnoho parametrů pro nastavení vzhledu. Z tohoto důvodu neuvádím syntaxi této funkce, ani popis všech parametrů. Popis těchto parametrů je na oficiálních stránkách nástroje RRDtool [27]. Pro představu uvedu pouze ukázku vytvoření základního grafu.

```
rrdtool graph vytizeniCPU.png
--title "Vytizeni CPU"
--vertical-label "Vytizeni [%]"
--start "now-1d"
--end "now"
--imgformat "PNG"
DEF:data=vytizeniCPU.rrd:cpu:AVERAGE
LINE1:data#00ff00:Vytizeni CPU
```

Výpis 7: Ukázka vytvoření grafu

6.6 Detekce anomálií v datech

Nástroj RRDtool obsahuje kromě konsolidačních funkcí, také specializované funkce, které umožní: predikci dat, vytvoření hranic spolehlivosti pro data a vyznačení anomálního chování v datech. K tomu využívá algoritmu Holt-Winters. Tyto specializované RRA se odlišují od skutečných konsolidačních funkcí (CF) několika způsoby. Každý RRA je aktualizován pro všechny PDP (Primary Data Point) a také jsou tyto RRA na sobě závislé. K vytvoření hranic spolehlivosti pro data v reálném čase je nutné nastavit RRA: SEASONAL, DEVSEASONAL, DEVPREDICT a dále musí být použit jeden z dvojce parametru HWPREDICT nebo MHWPREDICT.

Vyhlazené nebo předpovídané hodnoty jsou uložené v HWPREDICT nebo MHWPREDICT. HWPREDICT a MHWPREDICT jsou dvě metody pro výpočet Holt-Winters algoritmu a jsou navzájem zaměnitelné. Obě metody rozkládají data do tří složek: základ (baseline), lineární trend (sklon) a sezonní trend. HWPREDICT přičítá svůj sezonní koeficient k základu a vytvoří tak predikci hodnot. MHWPREDICT pro predikci hodnot vynásobí svůj sezonní koeficient se základem. Rozdíl mezi těmito dvěma metodami je patrný, když se výrazně mění základ v sezonním úseku. HWPREDICT bude předpovídat konstantní vývoj vzhledem k základu. MHWPREDICT bude předpovídat růst nebo klesání v poměru k základu. SEASONAL ukládá sezonní koeficienty s délkou jednoho cyklu. DEVSEASONAL ukládá sezonní odchylky s délkou jednoho cyklu. SEASONAL a DEVSEASONAL slouží pro výpočet Holt-Winters algoritmu. Pro každý sezonní úsek je pouze jeden záznam v sezonním cyklu. Při generování PDP, každých 5 minut a sezonním cyklu 1 den budou mít oba RRA SEASONAL a DEVSEASONAL 288 řádků.

Předpokládané odchylky jsou uloženy v DEVPREDICT (při převedení standardní odchylky i do záporných hodnot, získáme hranice spolehlivosti). RRA FAILURES si ukládá binární ukazatele, pokud v předchozím úseku dojde k překročení hranice spolehlivosti, nastaví tento ukazatel na hodnotu 1. [6, 28]

6.6.1 Parametry RRA pro detekci anomálií

Za účelem zjednodušení pro začínající uživatele RRDtool podporuje implicitní vytvoření zbylých čtyř RRA, pokud nadefinujeme HWPREDICT. Parametr *rra-num* v tomto případě nedefinujeme. Základní nastavení není příliš vhodné, proto je lepší vytvořit RRA (HWPREDICT, SEASONAL, DEVSEASONAL, DEVPREDICT a FAILURES) explicitně nebo pomocí příkazu *resize* změnit tyto hodnoty. [6, 28]

```
RRA:HWPREDICT:rows:alpha:beta:seasonal period[:rra−num]
RRA:MHWPREDICT:rows:alpha:beta:seasonal period[:rra−num]
RRA:SEASONAL:seasonal period:gamma:rra−num[:smoothing−window=fraction]
RRA:DEVSEASONAL:seasonal period:gamma:rra−num[:smoothing−window=fraction]
RRA:DEVPREDICT:rows:rra−num
RRA:FAILURES:rows:threshold:window length:rra−num
```

Výpis 8: Syntaxe RRA pro detekci anomálií

- *rows* - určuje počet hodnot, z kterých se bude vytvářet predikce. Pro HWPREDICT hodnota *rows* musí být větší, než hodnota *seasonal period*. Při implicitním vytvoření RRA DEVPREDICT je hodnota *rows* (DEVPREDICT) rovná hodnotě *rows* (HWPREDICT). Při implicitním vytvoření RRA FAILURES je hodnota *rows* (FAILURES) nastavená na stejnou hodnotu jako hodnota *seasonal period* (HWPREDICT).
- *seasonal period* - udává z kolika PDP se bude skládat sezonní úsek. Při implicitním vytvoření RRA SEASONAL a DEVSEASONAL je hodnota *seasonal period* automaticky nastavena na hodnotu *seasonal period* (HWPREDICT). Při explicitním vytvoření je potřeba ověřit, že všechny tyto hodnoty *seasonal period* jsou totožné. Hodnota musí být celočíselná a větší než 2.
- *alpha* - tento parametr slouží pro výpočet koeficientu *základ* (baseline) v Holt-Winters forecasting algoritmu. Hodnota tohoto parametru musí být v rozmezí 0 až 1. Samotný parametr určuje, jak velkou váhu budou mít nové pozorování pro vytváření predikce základní složky. Hodnota parametru blíž k 1 znamená, že nové pozorování mají větší váhu pro vytváření predikce základní složky. Hodnota parametru blíž k 0 znamená, že větší váhu pro predikci základní složky budou mít pozorování z minulosti.
- *beta* - tento parametr je použit pro výpočet koeficientu *lineární trend* (sklon) v Holt-Winters algoritmu. Hodnota tohoto parametru musí být v rozmezí 0 až 1. Samotný parametr určuje, jak velkou tendenci budou mít data pro lineární růst.
- *gamma* - tento parametr slouží pro výpočet koeficientu *sezonní trend* v Holt-Winters algoritmu (HWPREDICT), nebo může sloužit jako parametr přizpůsobení pro exponenciální vyhlazování odchylek. Hodnota tohoto parametru musí být v rozmezí 0 až 1. Při implicitním vytvoření RRA SEASONAL a DEVSEASONAL hodnota *gamma* bude nastavena stejnou hodnotu jako hodnota *alpha* (HWPREDICT). Při explicitním vytvoření RRA SEASONAL a DEVSEASONAL musí být hodnota *gamma* pro oba RRA totožná.

- *smoothing-window* - určuje část úseku, který by měl být zprůměrován kolem každého bodu (vyhlazování). Ve výchozím nastavení je parametr *smoothing-window* nastaven na 0.05. To znamená, že každá hodnota v SEASONAL a DEVSEASONAL může být někdy nahrazena zprůměrováním hodnot z nejbližších okolí (*seasonal period* * 0.05). Toto chování lze vypnout tím, že parametr *smoothing-window* nastavíme na 0.
- *rra-num* - RRA jsou na sobě závislé a pomocí parametru *rra-num* se vytváří mezi jednotlivými RRA vazby, jedná se tedy o index se základem 1. Pokud jsme definovali pouze RRA HWPREDICT a zbylé RRA byly vytvořeny implicitně, není potřeba tento parametr nastavovat. Při explicitním vytvoření všech RRA, je tento parametr velmi důležitý. Závislosti mezi jednotlivými RRA jsou zobrazeny zde:
  - HWPREDICT *rra-num* je index pro RRA SEASONAL.
  - SEASONAL *rra-num* je index pro RRA HWPREDICT.
  - DEVPREDICT *rra-num* je index pro RRA DEVSEASONAL.
  - DEVSEASONAL *rra-num* je index pro RRA HWPREDICT.
  - FAILURES *rra-num* je index pro RRA DEVSEASONAL.
- *threshold* - udává minimální počet překročení hranic spolehlivosti (pozorované hodnoty jsou mimo hranice spolehlivosti), při tomto překročení je zaznamenáno selhání. Při implicitním vytvoření RRA FAILURES je tento parametr nastaven na hodnotu 7.
- *window length* - udává počet časově závislých bodů v daném úseku. Hodnota parametru *window length* musí být celé číslo větší nebo rovno parametru *threshold* a zároveň musí být menší nebo rovno hodnotě 28. Při implicitním vytvoření RRA FAILURES je tento parametr nastaven na hodnotu 9.

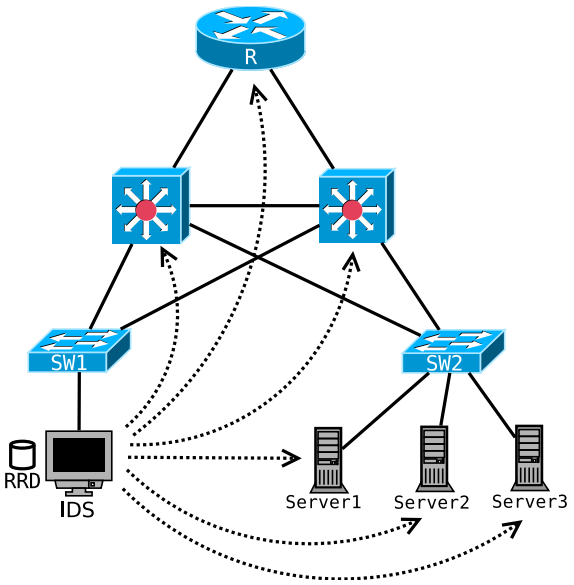
```
RRA:HWPREDICT:1440:0.1:0.0035:288:3
RRA:SEASONAL:288:0.1:2
RRA:DEVPREDICT:1440:5
RRA:DEVSEASONAL:288:0.1:2
RRA:FAILURES:1440:3:5:5
```

Výpis 9: Ukázka definice RRA pro detekci anomálií (1 den)

7 Návrh a použití systému

Systém, který jsem v této diplomové práci vytvořil, je založený na nástroji RRDtool a na protokolu SNMP. Jedná se o IDS (*Intrusion Detection System*) systém. Tento systém pracuje pod operačním systémem Linux. Pro práci s nástrojem RRDtool jsem zvolil programovací jazyk *Perl*. Tento programovací jazyk jsem použil hlavně kvůli knihovně *RRDs*, která umožňuje výpis chybových hlášení. To velmi pomohlo při hledaní chyb ve skriptech.

Mojí snahou bylo, aby tento systém mohl být využit i v malých sítích společně se zařízením Raspberry Pi. Systém nemá velké systémové požadavky. Jedná se tedy o systém, který může být použit v domácích a malých sítích. Není potřeba mít výkonný server pro toto domácí použití. Raspberry Pi je velmi úsporné zařízení s nízkou spotřebou a dostatečným výkonem pro tyto účely. Při využití v rozsáhlejších sítích je potřeba už výkonnější zařízení nebo rozdělit zátěž mezi několik zařízení.



Obrázek 18: Využití systému v rozsáhlejších sítích

IDS systém bude vystupovat v síti jako SNMP Manager (viz obrázek 18). Ostatní zařízení v síti se budou chovat jako SNMP agent. Pomocí SNMP bude systém získávat informace o daném zařízení v síti a ukládat je do své RRD databáze. RRDtool poté pomocí algoritmu Holt-Winters vytvoří predikce pro tyto data. Určí odchylky od vytvořeného modelu sítě, na základě kterých detekuje anomálie. Následně jsou tyto informace vykresleny v grafu pro přehlednější monitorování. Pro ještě lepší vizualizaci výsledků, běží na zařízení společně s monitorovacím systémem, také webový server, kde budou tyto grafy zobrazeny. Je tak možné lépe porovnávat závislosti v grafech a pozorovat, co všechno bylo ovlivněno. Systém automaticky spouští skripty pomocí nástroje Cron. Tyto skripty jsou jádro systému. Obsahují veškerou komunikaci a logiku.

Další klíčovou věcí pro detekci anomálií je nastavení hodnoty *threshold*. V základu je tato hodnota nastavena na 7. To znamená, že anomálie je v grafu vykreslena, až když 7 hodnot překročí hranice spolehlivosti. S krokem 1 minuta bude anomálie tedy vyznačena až po 7 minutách. S krokem 5 minut bude anomálie vyznačena až po 35 minutách. Nastavením velmi nízké hodnoty *threshold* může způsobit generování mnoho falešných poplachů *False Positive*. Záleží na požadavcích na systém. V některých případech může být požadavek na vyznačení každé anomálie, která nastane. V takovém případě parametr *threshold* nastavíme na hodnotu 1. Já jsem nastavil systém tak, aby vykreslil anomálii po třech překročeních hranic spolehlivosti. Hodnota *threshold* je tedy nastavena na 3. Některé hodinové grafy mají tento parametr nastavený na hodnotu 1.

Součásti návrhu bylo také rozhodnutí, jaké dlouhé časové úseky vykreslovat v grafech. Vytvořil jsem tedy grafy pro tři různě dlouhá časová období a na nich testoval detekci anomálií. Monitorování bylo rozděleno do časových úseků takto: za poslední hodinu, den a týden. Tyto časové období byly zvoleny tak, aby pokryly okamžité dění v síti, denní náhled a možnost porovnání v týdenním měřítku jednotlivé dny. Je možné vytvářet grafy i pro delší časové úseky (měsíc, rok), které slouží pro dlouhodobé monitorování. Tyto dlouhodobé grafy z časových možností nebylo možné vyzkoušet.

Nejvíce vypovídající byly grafy za poslední den. Ty dokáží celkem spolehlivě detekovat anomálie. Grafy, které vykreslovaly průběh za poslední hodinu také detekovaly anomálie, ale současně s tím docházelo k následnému ovlivnění průběhu. Po detekované anomálii docházelo k posunu hranic spolehlivosti a legitimní provoz byl poté označen za anomálii, protože byl mimo tyto hranice spolehlivosti. Výhoda hodinových grafů spočívá především v tom, že vykreslují aktuální stav sítě. Monitorování je skoro na úrovní *real-time*. Administrátor může rychleji zareagovat na danou událost, než u denních grafů.

Týdenní grafy se hodí spíše jen pro porovnávání mezi jednotlivými dny. Dokáží také detekovat anomálie, ale s velkým časovým zpožděním. Problém u dlouhodobých grafů spočívá v tom, z kolika hodnot by se měla výsledná hodnota skládat a kterou konsolidační funkci použít. Při použití průměru dochází ke zkreslení. Několik hodnot může být mimo hranice spolehlivosti, ale výsledná hodnota po zprůměrování spadá do intervalu spolehlivosti. Útok může být tedy zprůměrován s normálním provozem. Anomálie je v grafu přesto vyznačena, protože tento výpočet je prováděn pro každou hodnotu, ne pouze pro výslednou. Toto řešení není úplně vhodné, protože v grafu jsou zakresleny anomálie, i když nedošlo k překročení hranic spolehlivosti. Při použití jedné hodnoty pro výslednou hodnotu, dochází k tomu, že útok nemusí být zaznamenán vůbec. Protože je uložena pouze aktuální hodnota a poté databáze čeká dlouhý interval na další hodnotu. V tomto intervalu může být proveden útok a v grafu nebude zaznamenán. V některých případech může být řešením použití konsolidační funkce, která vybere maximální hodnotu a ta je uložena do databáze. Za anomálii nemůžeme, ale považovat pouze překročení horní hranice spolehlivosti, například zvýšený provoz na serveru. Systém musí být schopen detekovat i anomálie, které jsou mimo dolní hranici spolehlivosti. Snížený provoz na serveru, může být způsoben výpadkem části sítě a znepřístupnění tak tohoto serveru.

7.1 Ukázkový skript pro detekci anomálií

Výsledkem mé práce je také několik připravených skriptů pro detekci anomálií. Pro uživatele, kteří nemají moc velké zkušenosti s nástrojem RRDtool jsem vytvořil také ukázkový skript. Ten slouží pro jednoduché vytvoření vlastního skriptu pro detekci anomálií. Tento skript jsem se snažil vytvořit tak, aby uživatel do kódu musel zasahoval co nejméně. Ukázkový skript se skládá ze 4 částí: proměnné, vytvoření databáze, naplnění databáze a vykreslení grafu. Pro základní fungování stačí uživateli, aby nadefinoval proměnné na začátku skriptu. Ukázkový skript má nastavené hodnoty tak, aby monitoroval úsek jednoho dne. Je vytvořen pro práci s protokolem SNMP.

7.1.1 Ukázkový skript: 1. část (proměnné)

Na začátku každého skriptu se uvádí interpret pro jeho překlad. V druhém řádku je nadefinována knihovna RRDs, která byla vytvořena právě pro jazyk *Perl*. Dále už následují proměnné. Do první proměnné (start) se ukládá aktuální čas. Pokud není potřeba nějaké specifické chovaní, tuto proměnou neměníme. V následujících dvou proměnných je nastavena cesta a název pro RRD databázi a graf, kde se budou ukládat.

```
#!/usr/bin/perl
use RRDs;
my $start=time;
my $rrd="/home/rrdtool/rrdtool/skola/vytizeniCPU.rrd";
my $graf="/var/www/vytizeniCPU.png";
my $community="heslo";
my $IP="192.168.1.103";
my $OID=".1.3.6.1.2.1.5.1.0";
my $DS="cpu";
#my $DST="GAUGE";
#my $DST="COUNTER";
my $nadpis="Vytizeni␣CPU␣na␣Serveru";
my $osa="Vytizeni␣[%]";
my $vypis="CPU";
```

Výpis 10: Ukázkový skript: 1. část (proměnné)

Další tři proměnné slouží pro nastavení protokolu SNMP. Nejprve nastavíme community heslo pro SNMP. Potom definujeme IP adresu SNMP agenta, z kterého získáme informaci a dané OID, pod kterým tuto informaci nalezneme. Dále pak definujeme jméno pro Data Source, který se nastavuje při vytvoření databáze a odkomentujeme jeden z dvojce DST (Data Source Type). Vždy může být aktivní jen jeden, který DST použít, zjistíme například z programu MIB browser nebo ze základního výpisu příkazu *snmpget*. Před samotnou hodnotou je vypsaný datový typ. Poslední tři proměnné slouží pro nastavení grafu. Pro základní nastavení grafu stačí nadefinovat nadpis, popis osy *y* a výpis, který se bude zobrazovat ve spodní části pod grafem.

7.1.2 Ukázkový skript: 2. část (vytvoření databáze)

Před vytvořením databáze se vždy ověří jestli už daná databáze existuje, pokud neexistuje, vytvoří ji podle uvedených specifikací. Funkce CREATE slouží pro vytvoření databáze. Její základní parametry jsme již nadefinovali v předchozí části. Jediný parametr, který lze změnit je parametr *step*. Hodnota 300 udává, že databáze bude očekávat nové hodnoty každých 300 sekund (5 minut). Dále definujeme parametr DS (*Data Source*). Název a DST jsme již zvolili v předchozí části. Hodnota 600 určuje (*heartbeat*), jak dlouho bude databáze čekat na novou hodnotu. Pokud nová hodnota nedorazí do 600 sekund, do databáze se zapíše hodnota NAN (Not A Number). Parametry *Min* a *Max* jsou nastaveny na U (*UNKNOWN*). Tuto hodnotu nastavuje, pokud nevíme jakých hodnot může monitorovaná veličina nabývat. U vytížení procesoru můžeme nastavit tyto parametry na *0* a *100*. Dále už je pouze definování RRA (Round-Robin Archive).

```
if (not -f $rrd){
RRDs::create ($rrd,"--start",$start-1,"--step",300,
"DS:$DS:$DST:600:U:U",
"RRA:AVERAGE:0.5:1:2016",
"RRA:HWPREDICT:1440:0.1:0.0035:288:3",
"RRA:SEASONAL:288:0.1:2",
"RRA:DEVPREDICT:1440:5",
"RRA:DEVSEASONAL:288:0.1:2",
"RRA:FAILURES:1440:3:5:5"
);
my $ERROR=RRDs::error;
die "$0: Chyba pri vytvoreni databaze '$rrd':$ERROR\n" if $ERROR;
};
```

Výpis 11: Ukázkový skript: 2. část (vytvoření databáze)

První RRA je základní pro běžné monitorování. V našem případě konsolidační funkce bude vždy nastavena na *AVERAGE*, i když průměr z jedné hodnoty, je vždy ta daná hodnota. Následuje parametr *xff*, který je nastavený na hodnotu 0.5. Tím je určeno, kolik procent hodnot musí být validních pro sestavení výsledné hodnoty. Kolik hodnot pro sestavení výsledné hodnoty bude použito představuje další parametr (*steps*). V našem případě je nastavený na hodnotu 1. Poslední parametr pro základní RRA je *rows*. Tento parametr určuje velikost databáze. Kolik řádků bude databáze obsahovat. Pro jeden den je tato hodnota 1440 ($24 * 60$). Z důvodu zachování aspoň částečné historie dat je v našem případě tato hodnota nastavena na 2016. Další RRA se už vztahují k detekci anomálií. Zvolil jsem explicitní vytváření navazujících RRA na *HWPREDICT*. Hodnota *rows* 1440 určuje, z kolika hodnot se bude vytvářet predikce. V našem případě se jedná hodnoty za jeden den. Následují parametry adaptace. Ty jsem zvolil na základě doporučení [6], jehož autorem je Jake D. Brutlag. Parametr *seasonal period* je nastaven na hodnotu 288 ($1440/5$ minut). Jedná se o podíl hodnoty *rows* a hodnoty *step* v minutách. Další RRA jsou závislé a provázané s těmito hodnotami. Poslední RRA *FAILURES* slouží k zachycení selhání (anomálií). Důležitý parametry je *threshold*, který je nastavený na hodnotu 3. *Threshold* určuje kolik hodnot musí překročit hranice spolehlivosti, aby došlo k zaznamenání anomálie. Jestliže při vytváření databáze nastane chyba (špatně nadefinovaný

parametr), skript vypíše chybové hlášení do terminálu. To slouží pro rychlejší vyhledání chyby ve skriptu.

7.1.3 Ukázkový skript: 3. část (naplnění databáze)

Naplnění databáze se provádí jen v pár krocích. Většinu parametru nastavíte už v první části. Pokud hodnoty získáváte jinak, než pomocí SNMP protokolu, je potřeba správně definovat příkaz pro získání hodnoty. Do proměnné *v* se uloží hodnota získaná pomocí příkazu *snmpget*. V příkazu *snmpget* kromě už nastavených parametrů jsou ještě další dva parametry. Jedním určíme verzi SNMP protokolu. Ta je nastavena na SNMPv2 (*-v2c*). Druhý parametr slouží pro formátování výpisu. Nastavením *-Oqv* získáme pouze číselnou hodnotu. Nemusíme už provádět žádné vyfiltrování hodnoty z výpisu.

```
$v=(‘snmpget −v2c −c $community −Oqv $IP $OID‘);
chomp($v);
#print $v;
RRDs::update $rrd,"$start:$v";
```

Výpis 12: Ukázkový skript: 3. část (naplnění databáze)

Příkaz $chomp(\$v)$ odstraní nový řádek z výpisu. Bez tohoto příkazu by se do databáze neuložila získaná hodnota. Další příkaz slouží pro testování. Po odkomentování a spuštění skriptu v terminálu se vypíše získaná hodnota do terminálu. Slouží pro ověření zda získáváme správnou hodnotu. Nakonec pomocí funkce *UPDATE* se uloží hodnota do databáze. Všechny parametry této funkce jsou již nadefinovány, není potřeba provádět žádnou úpravu.

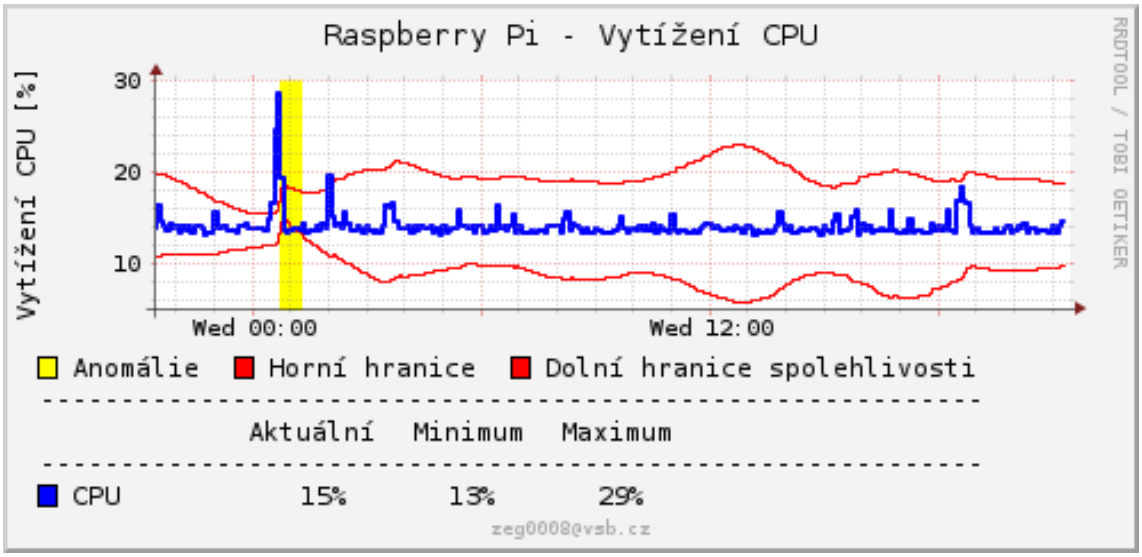
7.1.4 Ukázkový skript: 4. část (vykreslení grafu)

V poslední části se definuje nastavení grafu pro jeho vykreslení. Pro základní fungování stačí nastavit pouze proměnné na začátku skriptu. Funkce GRAPH má mnoho parametrů pomocí kterých lze graf všemožně upravovat. Více informací o těchto parametrech naleznete zde [27]. Pokud budeme chtít změnit délku monitorovacího úseku, musíme nejprve upravit 2. část skriptu a nastavit hodnoty pro námi nově zvolený úsek. V nastavení grafu musíme poté upravit parametr *start*. Společně s parametrem *end* nastavují časové rozmezí pro vykreslování hodnot v grafu. Mezi parametry, které je možné upravit patří *imgformat*. Já jsem zvolil generování grafu do formátu PNG. Je možné použít i jiné formáty (PDF, SVG nebo EPS). Následuje část (*DEF*) sloužící k provázání databáze s grafem. Monitorované hodnoty jsou načteny do proměnné *data*. V proměnné *pred* je uložená predikce hodnot. Proměnná *dev* obsahuje odchylku od predikce. Proměnná *fail* má v sobě uloženy informace o tom, zda došlo k překročení hranic spolehlivosti. Pro vytvoření hranic spolehlivosti využijeme proměnnou *pred* a přičteme/odečteme k ní dvojnásobek proměnné *dev*. *VDEF* jsem využil pro získaní minimální, maximální a aktuální hodnoty pro číselný výpis pod grafem. Barvu vykreslovaných informací můžeme změnit nastavením hodnoty RGB za # u funkcí: *TICK*, *LINE1* a *LINE2*. Poté už následuje jen definice číselného výpisu. Na konci skriptu je ověření, zda nedošlo k chybě při generování skriptu. Pokud ano, je vypsáno chybové hlášení o dané chybě do terminálu.

```
RRDs::graph"$graf",
"--title","$nadpis",
"--vertical-label","$osa",
"--watermark","zeg0008\@vsb.cz",
"--start","now-1h","--end","now",
"--imgformat","PNG",
"DEF:data=$rrd:$DS:AVERAGE",
"DEF:pred=$rrd:$DS:HWPREDICT",
"DEF:dev=$rrd:$DS:DEVPREDICT",
"DEF:fail=$rrd:$DS:FAILURES",
"CDEF:upper=pred,dev,2,*,+",
"CDEF:lower=pred,dev,2,*,-",
"VDEF:dataLast=data,LAST","VDEF:dataMin=data,MINIMUM","VDEF:dataMax=data,MAXIMUM",
"TICK:fail#ffff00 :1.0:Anomalie",
"LINE1:upper#ff0000:Horni hranice",
"LINE1:lower#ff0000:Dolni hranice spolehlivosti\\n",
"COMMENT:------------------------------------------------------------\\n",
"COMMENT:                Aktualni","COMMENT:  Minimum","COMMENT:  Maximum\\n",
"COMMENT:------------------------------------------------------------\\n",
"LINE2:data#0000ff:$vypis",
"GPRINT:dataLast:    %7.3lf","GPRINT:dataMin:   %7.3lf ","GPRINT:dataMax:   %7.3lf \\n",
;
if ($ERROR=RRDs::error){
print "ERROR: $ERROR\n";
};
```

Výpis 13: Ukázkový skript: 4. část (vykreslení grafu)

Výsledkem tohoto skriptu může být například tento graf:



Obrázek 19: Ukázkový graf - vytížení CPU

7.2 Instalace a konfigurace systému

Následující postup instalace a konfigurace je uvedený pro operační systém Linux a distribuce vycházející z Debian. Je nutné mít připojení k internetu. Instalace probíhá z repositářů dané distribuce. Instalace a konfigurace byla otestována na těchto operačních systémech: Raspbian (2014-08), Raspbian (2015-02) a Linux Mint (13.10). Systém by měl být funkční i na jiných distribucích, než jen na těch, které vycházejí z Debian. Tato funkčnost, ale nebyla ověřena a postup instalace může být odlišný.

7.2.1 Instalace RRDtool

Pro správné fungování nástroje RRDtool a připravených skriptů je potřeba nainstalovat dvě věci. RRDtool a knihovnu *librrds-perl* pro Perl. Samotný nástroj a knihovnu nainstalujeme pomocí tohoto příkazu:

```
sudo apt-get install rrdtool librrds-perl
```

Není už potřebná žádná další konfigurace tohoto nástroje. Pokud budete využívat vlastní skripty a jiný jazyk pro jejich tvorbu, není nutná instalace knihovny *librrds-perl*. Pro nainstalovaní poslední verze RRDtool, je potřeba tuto verzi stáhnout z oficiálních stránek. Tato instalace je o něco složitější, protože je potřeba stažené kódy nejprve zkompilovat.

7.2.2 Instalace webového serveru

Instalace webového serveru je velmi jednoduchá. Vyzkoušel jsem dvě verze webového serveru: *Apache* a *Lighttpd*. Pro Raspberry Pi doporučuji spíše *Lighttpd*. Jedná se o rychlý a paměťově nenáročný webový server. Může být použit ale i *Apache*. Pro základní chod není potřebná další konfigurace, stačí jen provést instalaci:

```
sudo apt-get install apache2
```

nebo

```
sudo apt-get install lighttpd
```

Pro ověření funkčnosti webového serveru stačí zadat do webového prohlížeče IP adresu zařízení nebo *localhost*. IP adresa může být ve verzi 4 nebo ve verzi 6. Na přiloženém CD je šablona pro webové stránky.

7.2.3 Instalace a konfigurace SNMP

Pro nainstalování SNMP managera zadejte do terminálu tento příkaz:

```
sudo apt-get install snmp
```

Pro nainstalování SNMP agenta zadejte do terminálu tento příkaz:

```
sudo apt-get install snmpd
```

Nyní je nutné provést konfiguraci SNMP agenta. V základní konfiguraci SNMP agent naslouchá pouze na adrese *localhost* a vypisuje informace pouze z větve system. Dále je SNMP agent nastavený pouze pro provoz na IPv4 a využívá SNMP ve verzi 1. Je nutné proto upravit soubor *snmpd.conf*, který se nachází v adresáři */etc/snmp/*. Úpravu je možné provést například pomocí nástroje *nano* nebo *vim*. Následující výpis ukazuje, které řádky je potřeba upravit:

```
#agentAddress  udp:127.0.0.1:161
agentAddress   udp:161,udp6:161
#rocommunity  public  default  -V systemonly
rocommunity heslo default
###pro IPv6###
rocommunity6 heslo default
```

V tomto nastavení bude SNMP agent naslouchat na všech IPv4 a IPv6 adresách, které má nakonfigurované. *Community-string* pro autentizaci byl nastaven na **heslo** a je využíváno SNMP ve verzi 2. Místo **heslo** můžete zvolit jakýkoliv jiný vámi zvolený řetězec. Všechny provedené změny se projeví až po restartování služby *snmpd*. To povedeme pomocí příkazu: **sudo service snmpd restart**. Pro otestování zadejte tyto příkazy do terminálu:

```
snmpwalk -v2c -c heslo IP_adresa_agenta
snmpget -v2c -c heslo IP_adresa_agenta .1.3.6.1.2.1.1.5.0
snmpget -Oqv -v2c -c heslo IP_adresa_agenta .1.3.6.1.2.1.1.5.0
```

Další zařízení, které jsem použil pro mojí diplomovou práci byly zařízení (směrovač, přepínač) od společnosti Cisco. Základní konfigurace SNMP agenta na těchto zařízeních je poměrně jednoduchá. Stačí v globálním konfiguračním režimu zadat tento příkaz:

```
snmp-server community heslo ro
```

7.2.4 Automatické spouštění skriptů

Velmi důležitou částí je i automatické spouštění skriptů. Pro tuto činnost jsem si vybral nástroj *Cron*. Tento nástroj je už součástí operačního systému, který používáme. Před samotným nastavením tohoto nástroje je potřeba změnit práva vytvořeným skriptům. To provedeme v terminálu tímto příkazem: `sudo chmod a+x nazevSkriptu.pl`

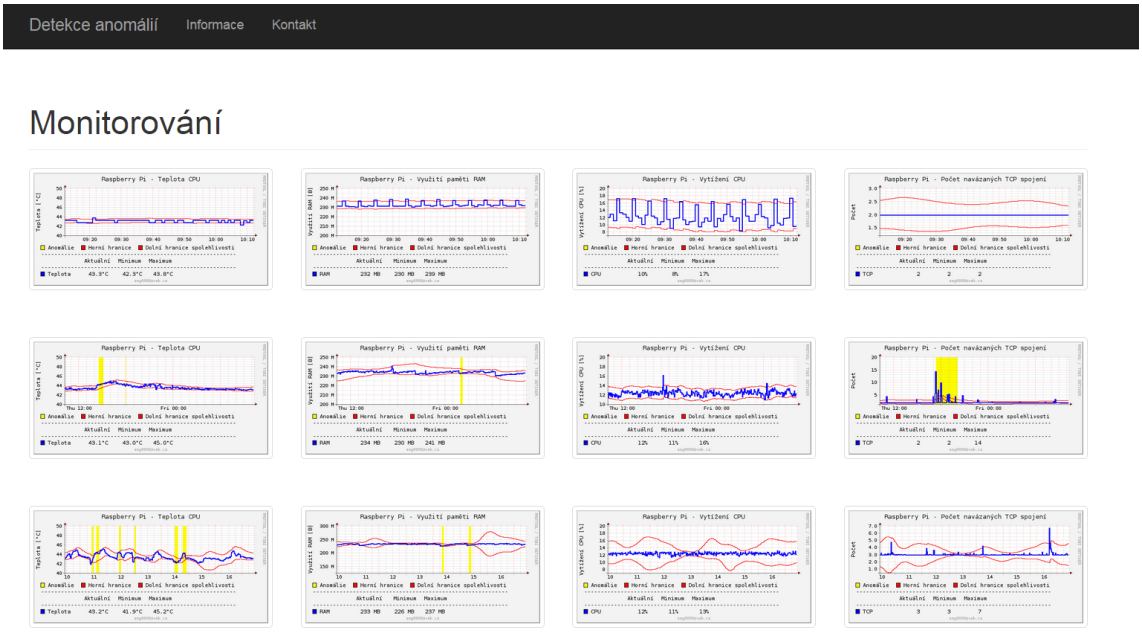
Skript je nyní spustitelný a je možné provést nastavení nástroje *Cron*. Skripty, které jsem vytvořil je potřeba spouštět s právy **root**, protože výsledný graf se ukládá do adresáře */var/www/*. Pokud budete ukládat výsledné grafy do domovského adresáře, není potřeba skript spouštět s těmito právy. Pro nastavení nástroje Cron, zadejte tento příkaz: `sudo crontab -e`. Nyní je nutné nadefinovat skripty, které se mají spouštět.

```
* * * * * /home/pi/rrdtool/Skript1.pl
*/5 * * * * /home/pi/rrdtool/Skript2.pl
```

Nejprve definujeme čas spuštění a poté úplnou cestu k našemu skriptu. *Skript1* je spouštěn každou minutu. *Skript2* je spouštěn každých 5 minut. Takto je nutné nadefinovat všechny skripty.

7.3 Zobrazení na webovém serveru

Součástí této práce je také šablona pro webové stránky. Tuto šablonu stačí nakopírovat do adresáře */var/www/*. Webové stránky slouží pro lepší vizualizaci a přehlednost celého monitorování. Lépe se tak určují závislosti mezi jednotlivými monitorováními, než při procházení jednoho grafu za druhým. Vizualizace na webovém serveru je zobrazena na obrázku 20. Je možné využít vlastní webové stránky a nebo jiný způsob pro výslednou vizualizaci. Stránky jsou součástí CD, které je přiloženo k diplomové práci.



Obrázek 20: Ukázka zobrazení na webovém serveru

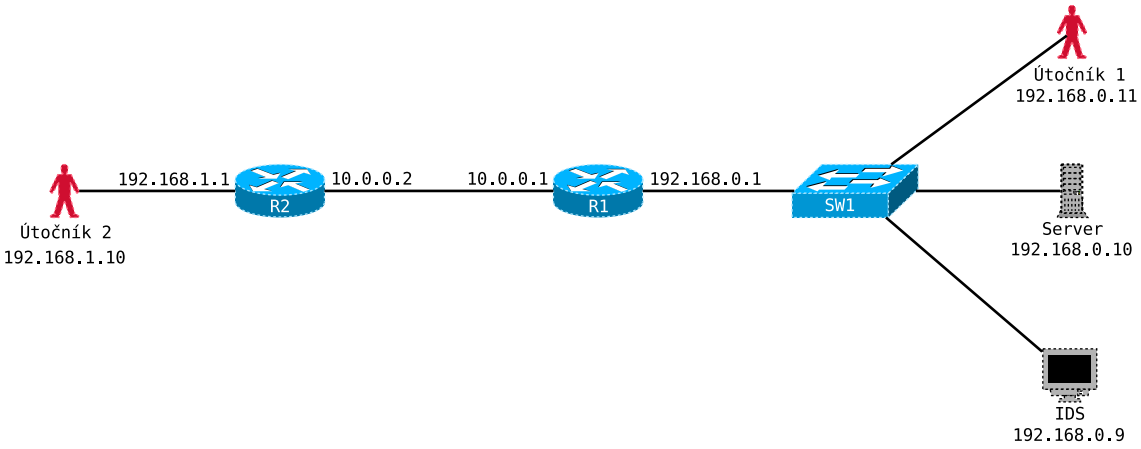
8 Testování systému

Testování tohoto systému bylo rozděleno do tří částí. Nejprve jsem systém testoval v programu GNS3, který slouží pro emulaci sítě. Druhá část testování probíhala ve školní laboratoři, kde Raspberry Pi, bylo připojeno ke školnímu přístupovému přepínači. Poslední část testování probíhala na mém domácím Raspberry Pi, které bylo připojeno přímo k internetu. Byly také testovány skripty pro různě dlouhé časové úseky. Se zaměřením na poslední hodinu, den a týden. Systémové informace, které jsem monitoroval byly: vytížení procesoru, teplota procesoru nebo třeba využití paměti RAM. Síťové informace jsem monitoroval tyto: počet navázaných TCP/UDP spojení, příchozí/odchozí provoz na rozhraní a čítače zaměřené na protokoly ICMP, TCP a UDP.

8.1 Testování v GNS3

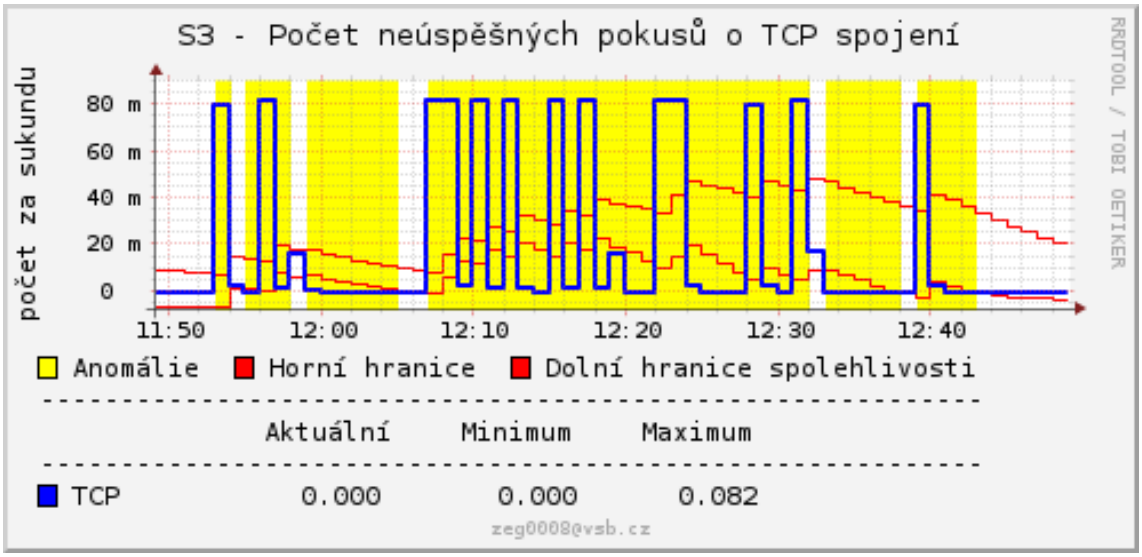
Program GNS3 umožňuje emulaci síťových zařízení a propojení nasimulované sítě s reálnou sítí. Tento program je velice užitečný pro testování. Lze v něm nakonfigurovat a vyzkoušet velmi mnoho technologií a různých topologií. Je velmi vhodný pro studijní účely a má více možností využití než *Cisco Packet Tracer*. [29]

V této části probíhalo základní testování nástroje RRDtool a protokolu SNMP. Získaní hodnoty ve vhodném formátu a ověření, zda se hodnota správně ukládá do databáze. Dále zde také proběhla zkouška prvních skriptů pro detekci anomálií a testování nastavených parametru. S tím souvisí i výběr informací pro monitorování. Vybíral jsem informace, které by bylo vhodné monitorovat pro detekci anomálií. Většina těchto informací jsou čítače jednotlivých protokolů. Tyto získané informace jsem poté testoval, jak dochází k jejich ovlivnění různým typem provozu. Pro tyto účely jsem vytvořil jednoduchou topologii, která je zobrazena na obrázku 21.

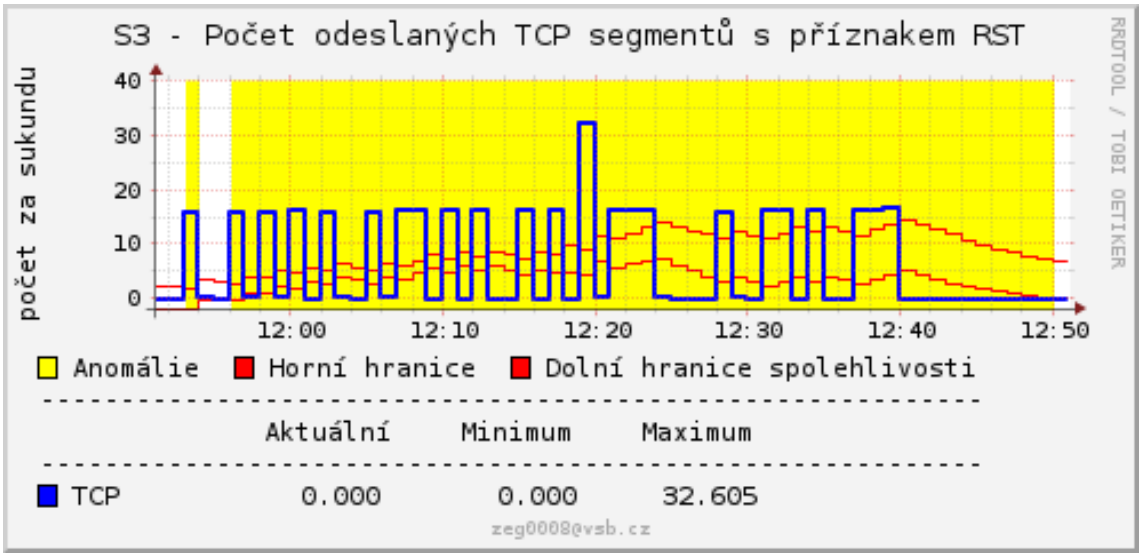


Obrázek 21: Testovací topologie v programu GNS3

Jednou z věcí, na kterou jsem se zaměřil je detekce skenování sítě. K tomuto účelu jsem využil nástroj Nmap. Tento nástroj umožňuje provádět různé typy skenování sítě. Zjišťoval jsem závislosti mezi jednotlivými typy skenování a jednotlivými čítači protokolů. Testoval jsem zda některý z čítačů dokáže zachytit jednotlivá skenování. Čítačů jsem monitoroval několik, protože jednotlivá skenování sítě mají různý průběh.



Obrázek 22: Detekce skenování sítě (1)



Obrázek 23: Detekce skenování sítě (2)

V grafech (viz obrázek 22 a 23) je vidět kdy probíhalo skenování sítě a tento útok je označen za anomálii. Toto skenování sítě ovlivnilo hranice spolehlivosti a legitimní

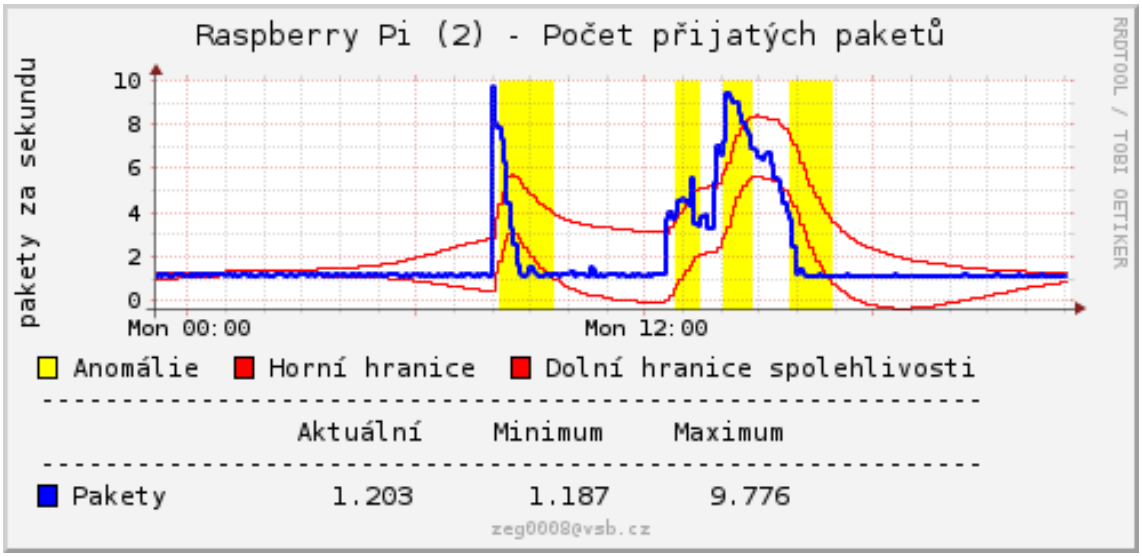
provoz byl také označen za anomálii. V grafech je velmi dobře vidět, jak se útoky začaly postupně stávat součástí predikce. Při delším časovém úseku by tyto útoky už nebyly detekovány jako anomálie. Z tohoto důvodu systém používá i grafy s delším časovým úsekem (den, týden), kde takovýto útok bude detekován jako anomálie.

Většinu typů skenování čítače opravdu zachytily. Jedná se ale pouze o laboratorní výsledky, a ve skutečném provozu toto skenování sítě nemusí být odhaleno. Nebylo bohužel možné toto chování ověřit ve skutečné síti. Skenování sítě je možné vysledovat na těchto grafech: Počet resetovaných TCP spojení, Počet odeslaných TCP segmentů s příznakem RST, Počet pasivně otevřených TCP spojení, Počet přijatých UDP datagramů s nedefinovaných portem a Počet neúspěšných pokusů o TCP spojení.

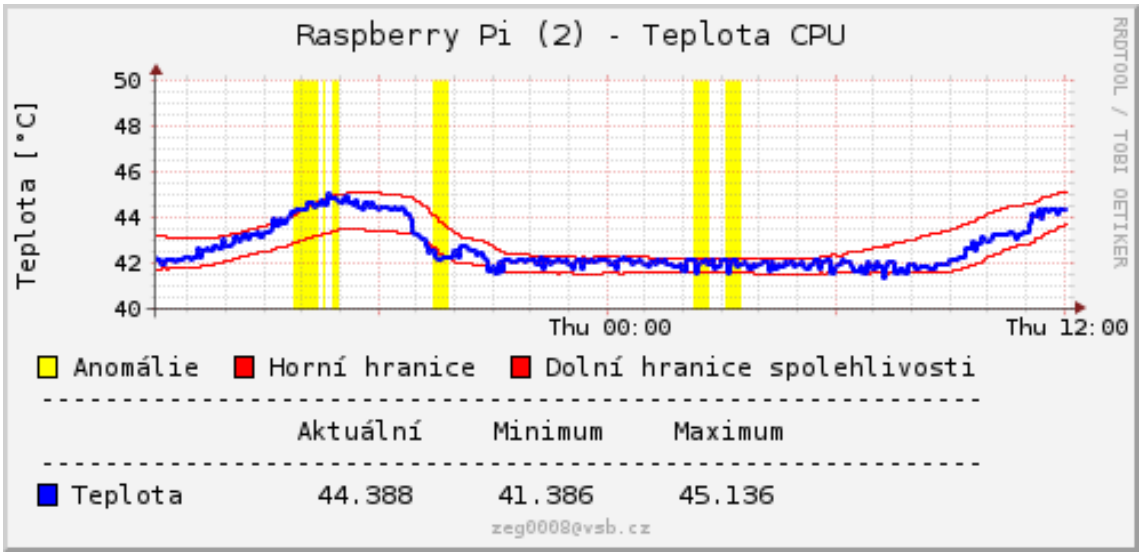
8.2 Testování ve školní laboratoři

V této části testování bylo zkoumáno zda dochází k anomáliím na zařízení, které je připojeno ke školní síti. Raspberry Pi bylo připojeno ke školnímu přístupovému přepínači v laboratoři počítačových sítí. Bylo nutné nastavit IPv6 adresu školního DNS serveru a povolit na Raspberry Pi modul pro IPv6, protože tato laboratoř je připojena k síti pouze prostřednictvím IPv6 protokolu. Zařízení získalo IPv6 adresu ze školního DHCP serveru a bylo tedy přístupné v rámci školní sítě nebo přes VPN.

Z bezpečnostních důvodů nebyl povolen přístup ke školnímu přístupovému přepínači a získávání informací pro monitorování přes SNMP přímo z něj. Proto data pro monitorování byla získávána pouze ze samotného Raspberry Pi, které bylo připojeno do školní sítě. Jednalo se tedy o *self-monitoring*. Grafy byly zaměřeny na časový úsek jednoho dne a byly průběžně zálohovány. Tento časový úsek byl zvolen zejména kvůli dobrým výsledkům v předchozím testování. Při tomto testování jsem nevytvářel žádný umělý provoz na zařízení.



Obrázek 24: Počet přijatých paketů v době výuky



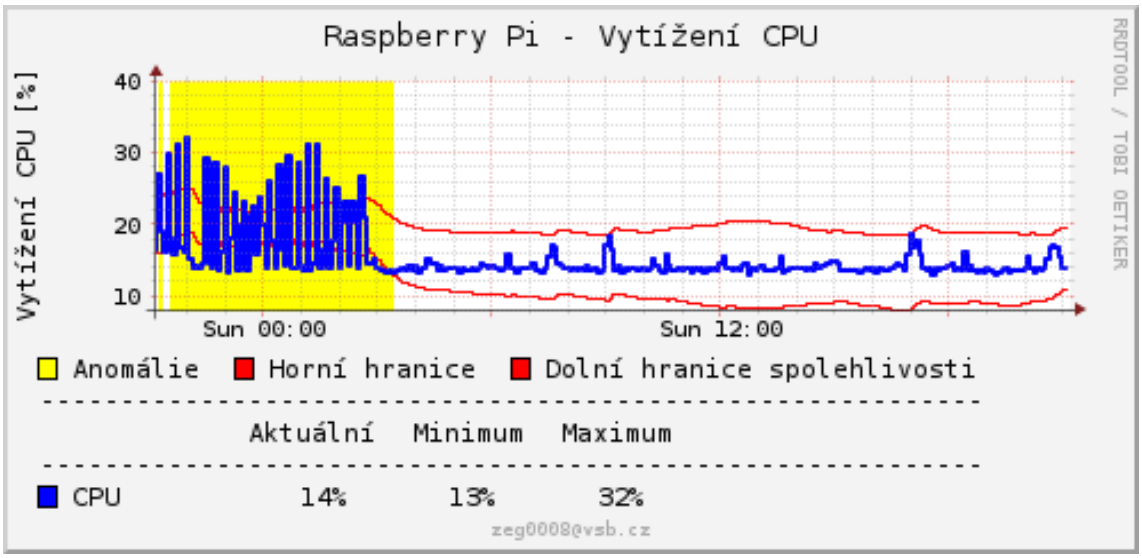
Obrázek 25: Teplota CPU ve školní laboratoři

Toto testování probíhalo několik týdnů. Očekával jsem, že se na tomto zařízení budou objevovat anomálie především v čase, kdy probíhá výuka. V grafech bylo možné vypozorovat, kdy výuka probíhá v nárůstu provozu. Byly zaznamenány změny hlavně v počtu přijatých paketů (viz obrázek 24), bajtů a teplotě procesoru (viz obrázek 25). Anomálie byly většinou zakresleny u grafu, který zobrazoval teplotu procesoru. Zařízení bylo sice umístěné v racku, ale bez žádného větrání. Při zapnutí zařízení (směrovače, přepínače) v ostatních racích docházelo k nárůstu teploty v racku, kde bylo umístěné monitorovací zařízení. Anomálie, které byly zaznamenány na zařízení připojené ke školní síti, můžeme tedy zařadit do kategorie (*False Positive*). V tomto testování se jednalo o falešné poplachy. V reálné síti by však zvýšená teplota v racku mohla znamenat nějaký problém a detekována anomálie by tedy mohla patřit do kategorie (*True Positive*). Jedna z anomálií, která se v tomto testování objevovala byla při zálohovaní dat. To jsem většinou prováděl ve večerních hodinách. Systém z předchozích naměřených hodnot očekával provoz jen v průběhu dne. Tato anomálie také patří do kategorie (*False Positive*), protože jsem na zařízení přistupoval osobně. Pokud by se v nočních hodinách snažil přistoupit někdo jiný na toto zařízení jednalo by se o anomálii (*True Positive*). Výsledky tohoto testování nebyly nikterak překvapivé. Na segmentu sítě ke kterému bylo připojeno monitorovací zařízení nedocházelo k neobvyklému provozu. Očekával jsem sice, že budou detekovány anomálie ve větším množství v čase výuky, ale nestalo se tak.

8.3 Raspberry Pi připojené k internetu

Největší vypovídající hodnotu má toto testování, protože Raspberry Pi bylo vystavené reálnému provozu na internetu. Předchozí dvě části testování byly spíše laboratorní. Pro tyto účely testování jsem si zakoupil vlastní doménu a nasměřoval jí na mé domácí Raspberry Pi. Toto testování jsem rozdělil ještě do dvou fází. V první fázi bylo Raspberry

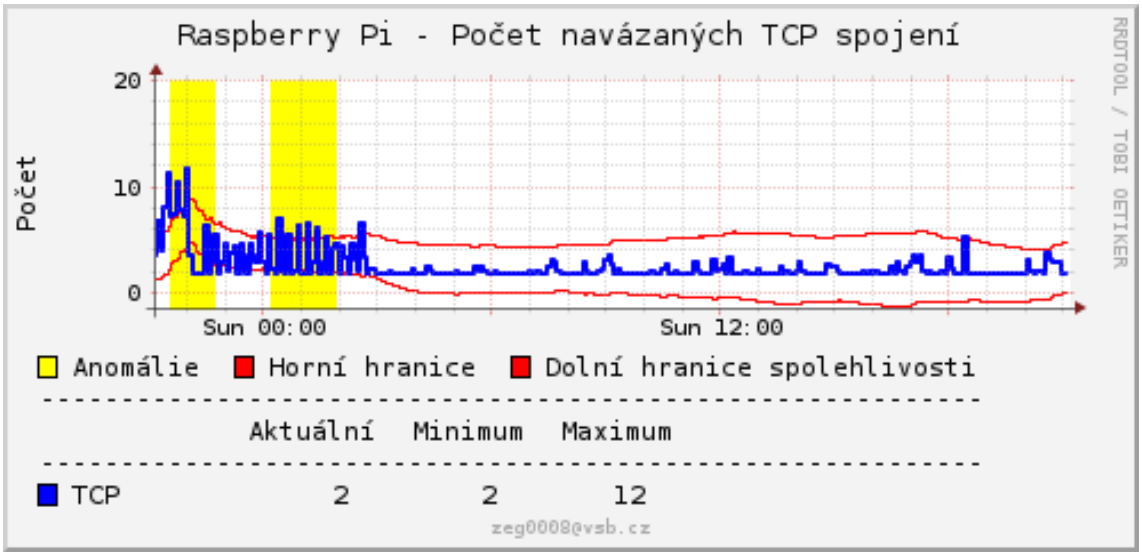
Pi připojeno k internetu bez jakéhokoliv omezení provozu nebo portů. V druhé fázi jsem připojil k internetu Raspberry Pi, které mělo přesně nadefinované pravidla pro provoz. Na tomto zařízení nebyly spuštěny žádné další služby, než kromě těch, které spouští samotný systém a ty které jsem nainstaloval pro monitorování. Na webovém serveru byly vykreslovány pouze grafy pro monitorování. Informace, které jsem monitoroval na tomto zařízení jsou: teplota procesoru, vytížení procesoru, využití paměti RAM a počet navázaných TCP spojení. Pro všechny tyto monitorované informace jsem vytvořil grafy, které zobrazovaly průběh za poslední hodinu, den a týden.



Obrázek 26: Vytížení CPU - detekce útoku

V první fázi tohoto testu jsem již na druhý den začal pozorovat, že v grafech dochází k anomáliím. Frekvence těchto anomálií se začala postupně zvyšovat. Po týdenním provozu byly tyto anomálie součástí každodenního monitorování. Po prozkoumání výpisů z logů jsem zjistil, že mé domácí Raspberry Pi se stalo terčem útoků. Útoky byly mířené na všechny otevřené porty a především na port 22. Na kterém běží deamon služby SSH, která slouží pro vzdálený přístup k zařízení nebo pro přenos souborů. Útoky pocházely z různých IP adres. Kromě toho, že se zvyšoval počet útoků, rostla také délka těchto útoků. Nejdelší zaznamenaný útok trval přes 24 hodin. Po tomto útoku nastala chyba SD karty a Rasberry Pi jsem musel přeinstalovat.

Tyto útoky jsem rozpoznal v grafech, které zobrazovaly vytížení procesoru (viz obrázek 26) a počet navázaných TCP spojení (viz obrázek 27). Mezi těmito dvěma grafy bylo možné vypozorovat souvislost, že se zvýšeným počtem TCP spojení došlo i ke zvýšení vytížení procesoru. Anomálie byly zaznamenány především u vytížení procesoru. Graf pro TCP spojení byl ovlivněn předešlými útoky. Tyto předešlé útoky ovlivnily hranice spolehlivosti a ty se díky tomu rozšířily. Proto tento útok v grafu pro počet navázaných TCP spojení nebyl označen po celou dobu jako anomálie.



Obrázek 27: Počet TCP spojení při útoku

Před druhou fází toho testu jsem nejprve Raspberry Pi částečně zabezpečil. Nadefinoval jsem pravidla pro filtrování provozu. Z internetu byl povolen přístup pouze na webové stránky. Z mé místní sítě jsem povolil přístup přes SSH. Protože jsem využíval SNMP, vytvořil jsem pravidlo pro povolení SNMP provozu pouze na adrese localhost. Všechen ostatní provoz byl zahazován. Toto zabezpečení jsem nastavil pomocí nástroje TCP Wrapper. Pravidla pro filtrování provozu se zapisují do dvou souborů: *hosts.allow* a *hosts.deny*. Tyto soubory se nacházejí v adresáři */etc/*.

hosts.allow:

```
httpd: ALL
sshd: 192.168.
snmpd: 127.0.0.1
```

hosts.deny:

```
ALL: ALL
```

Po vytvoření těchto pravidel jsem Raspberry Pi opět připojil k internetu. Provoz se ustálil a k útokům už přestalo docházet. Výsledkem tohoto testu je kromě detekce anomálií i zjištění proč docházelo k útokům. Pokud je k internetu připojený server nebo jakékoliv jiné zařízení, které není zabezpečené, stane se po krátké době terčem útoků. Nejedná se o útoky jednotlivců, ale o automatizované roboty (*botnet*). Tito roboti vyhledávají nezabezpečená zařízení na internetu a utočí na ně. Útoky jsou především směrovány na porty, na kterém dané zařízení naslouchá. Proto je nutné zařízení zabezpečit. Je dobré nadefinovat aspoň pravidla pro filtrování provozu nebo použít nějaký firewall. Tyto kroky ovšem server zabezpečí jen částečně.

9 Závěr

Cílem této diplomové práce bylo popsat typy síťových útoků a anomálií a popsat také formát souboru PCAP. Hlavním cílem této diplomové práce bylo navrhnout systém pro vyhledávání síťových anomálií na základě síťového provozu a systémových informací získaných z různých zařízení. Následně tento systém otestovat v laboratoři počítačových sítí a ověřit jeho funkčnost.

Systém, který jsem navrhl a vytvořil patří do kategorie IDS. Jádro tohoto systému je tvořeno protokolem SNMP a nástrojem RRDtool. Pro sběr informací jsem zvolil protokol SNMP a to z důvodu jeho široké podpory mnoha zařízeními a také kvůli množství informací, které se dají pomocí něho získat. Nástroj RRDtool slouží pro ukládání a zpracovávání získaných informací. RRDtool jsem zvolil z několika důvodů. Pro ukládaní dat využívá Round-Robin Database. Jedná se o databázi, která je konstantní a nezvětšuje se s nově příchozími daty. Pomocí funkce GRAPH je možné data vykreslovat přímo do grafu. Hlavním důvodem pro zvolení RRDtool bylo, že má v sobě implementovanou metodu Holt-Winters, která slouží pro predikci vývoje časové řady. Pro práci s nástrojem RRDtool jsem použil skriptovací jazyk Perl. Perl jsem zvolil zejména kvůli knihovně RRDs, která umožňuje výpis chybových hlášení do terminálu. To urychluje nalezení chyby ve skriptu. Pro vizualizaci výsledků monitorování jsem zvolil webový server. Takto je možné lépe sledovat závislosti mezi jednotlivými monitorováními a je tak zajištěna také lepší přehlednost.

Vytvořil jsem několik skriptů pro detekci anomálií, které jsou součástí této práce. Tyto skripty jsou rozděleny podle délky úseku monitorování. Nejkratší časový úsek je jedna hodina, poté následuje úsek pro jeden den a nejdelší úsek je pro jeden týden. Vytvořil jsem také jeden ukázkový skript. Ten slouží pro uživatele, kteří nemají příliš velké zkušenosti s nástrojem RRDtool. Na začátku tohoto skriptu stačí pouze nastavit nadefinované proměnné a tím si vytvoří skript, který dokáže detekovat anomálie v zadaných datech v rámci jednoho dne.

Testování mého IDS systému jsem rozdělil do tří částí: testování v programu GNS3, testování ve školní laboratoři počítačových sítí a pomocí zařízení Raspberry Pi připojené k internetu. V programu GNS3 jsem testoval základní funkčnost nástroje RRDtool a sběr informací pomocí SNMP. Testoval jsem zde první skripty pro detekci anomálií a následně jsem zde také testoval jak systém dokáže detekovat skenování sítě nástrojem Nmap. Druhá část testování probíhala ve školní laboratoři, kde zařízení Raspberry Pi bylo připojeno ke školnímu přepínači. Cílem tohoto testování bylo zjistit, zda dochází k anomáliím na zařízení, které je připojené ke školní síti. Výsledkem tohoto testování bylo, že k anomáliím dochází v době výuky. Jedná se, ale o anomálie typu *False Positive*. Nejednalo se o útoky na toto zařízení. Poslední testování probíhalo na mém domácím Raspberry Pi, které bylo připojené k internetu. Toto testování systém prověřilo nejlépe z hlediska funkčnosti. Testování jsem ještě rozdělil do dvou fázi. V první fázi zařízení bylo připojené k internetu bez pravidel pro filtrování provozu a v druhé fázi s těmito pravidly. V první fázi již druhý den testování systém vykresloval anomálie. Tyto anomálie se objevovaly stále častěji a jejich délka se prodlužovala. Po prozkoumání logů jsem zjistil, že se jedná o neoprávněné přístupy na toto zařízení. Zařízení se tedy stalo terčem

mnoha útoků z různých IP adres. Nejdelší zaznamenaný útok byl v délce okolo 24 hodin. Systém dokázal detekovat reálné útoky vedené přímo z internetu. V druhé fázi jsem tedy nastavil pravidla pro filtrování provozu. Z internetu byl dostupný pouze webový server. Přístup přes SSH byl povolený pouze z lokální sítě a na localhostu bylo povolené SNMP. Všechen ostatní provoz byl zahazován. Tyto pravidla zajistily základní ochranu před útoky a v grafech poté už žádné nové útoky nebyly zaznamenány.

Výhoda tohoto systému spočívá v jeho dynamičnosti. Nemá pevně zadané hodnoty, ale vytváří si predikce přímo z provozu v reálné síti. Může být tedy nasazen v různých topologiích. Další výhodou jsou systémové nároky na zařízení, na kterém běží tento IDS systém. Pro základní monitorování lze použít i zařízení Raspberry Pi. Tím je zajištěno, že tento systém může být využit i v menších sítích. Mezi výhody patří i okamžitá vizualizace výsledků monitorování pomocí grafů. Nevýhodou tohoto systému je, že může generovat poměrně mnoho falešných poplachů (False Positive). Nově příchozí útok nemusí být zaznamenán, pokud se předchozí útoky stanou součástí predikce. Další nevýhodou je, že tento systém už nemůže být dále rozšířen na IPS systém.

Systém, který vznikl v této práci může být využit jako doprovodný IDS systém pro monitorování a detekci útoků. Zejména kvůli okamžité vizualizaci. Uplatnění najde, ale také v domácích nebo malých sítích, kvůli svým systémovým nárokům. V testování se systém osvědčil, že dokáže detekovat útoky i na domácí webový server. Výsledky této práce mohou být také použity při výuce některého z předmětů, který se zabývá problematikou bezpečnosti v počítačových sítích nebo samotným monitorováním sítí. Dále je možné tuto práci rozšířit o statistický rozbor. S jakou přesností dokáže systém jednotlivé útoky spolehlivě detekovat a případně provést výpočet nových hodnot pro parametry adaptace u metody Holt-Winters.

Díky této diplomové práci jsem získal nové znalosti v počítačových sítí. Zejména v problematice bezpečnosti v počítačových sítích a samotného monitorování a detekci útoků. Dále jsem si rozšířil znalosti ve virtualizaci a práci s operačním systémem Linux. Při zpracovávání této práce jsem mohl uplatit znalosti získané v kurzu Cisco Academy. Vyzkoušel jsem si také, jak zabezpečit vlastní server proti neoprávněnému přístupu z internetu.

10 Reference

[1] RYBIČKA, Jiří *Latex pro začátečníky* 3. vyd. Brno: Konvoj, 2003, 238 s. ISBN 80-730-2049-1

[2] DEAL, Richard A *CCNA Cisco certified network associate study guide (exam 640-802)* New York: McGraw-Hill, 2008, 983 s. ISBN 0071497315

[3] SEDLÁK, Jan *Český internet je v pohotovosti proti DDoS útokům [online]*, [cit. 2015-04-9]. Dostupné z: http://connect.zive.cz/clanky/cesky-internet-je-v-pohotovosti-proti-ddos-utokum/sc-320-a-177132/default.aspx

[4] SHIREY, R. *RFC 4949 - Internet Security Glossary, Version 2 [online]*, [cit. 2015-04-9]. Dostupné z: https://tools.ietf.org/html/rfc4949

[5] SYMANTEC.COM *Security 1:1 - Part 3 - Various types of network attacks [online]*, [cit. 2015-04-9]. Dostupné z: http://www.symantec.com/connect/articles/security-11-part-3-various-types-network-attacks

[6] BRUTLAG, Jake D. *Aberrant Behavior Detection in Time Series for Network Monitoring [online]*, [cit. 2015-04-9]. Dostupné z: https://www.usenix.org/legacy/events/lisa00/full_papers/brutlag/brutlag_html/

[7] NMAP.ORG *Port Scanning Techniques [online]*, [cit. 2015-04-9]. Dostupné z: http://nmap.org/book/man-port-scanning-techniques.html

[8] WHITAKER Andrew, NEWMAN Daniel *Penetration Testing and Network Defense: Performing Host Reconnaissance [online]*, [cit. 2015-04-9]. Dostupné z: http://www.ciscopress.com/articles/article.asp?p=469623&seqNum=3

[9] NESSUS.ORG *Nessus Executive Report [obrázek]*, [cit. 2015-04-9]. Dostupné z: http://www.nessus.org/images/NessusExecutiveReport.png

[10] WIRESHARK.ORG *What is Wireshark? [online]*, [cit. 2015-04-9]. Dostupné z: https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

[11] WIRESHARK.ORG *Libpcap File Format [online]*, [cit. 2015-04-9]. Dostupné z: https://wiki.wireshark.org/Development/LibpcapFileFormat

[12] US-CERT *Understanding Denial-of-Service Attacks [online]*, [cit. 2015-04-9]. Dostupné z: https://www.us-cert.gov/ncas/tips/ST04-015

[13] PATRIKAKIS Charalampos, MASIKOS Michalis, ZOURARAKI Olga *Distributed Denial of Service Attacks [online]*, [cit. 2015-04-9]. Dostupné z: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-4/dos_attacks.html

[14] DuPaul, Neil *Spoofing Attack: IP, DNS & ARP [online]*, [cit. 2015-04-9]. Dostupné z: http://www.veracode.com/security/spoofing-attack

[15] SYMANTEC.COM *Security 1:1 - Part 1 - Viruses and Worms [online]*, [cit. 2015-04-9]. Dostupné z: http://www.symantec.com/connect/articles/security-11-part-1-viruses-and-worms

[16] CISCO.COM *What Is the Difference: Viruses, Worms, Trojans, and Bots? [online]*, [cit. 2015-04-9]. Dostupné z: http://www.cisco.com/web/about/security/intelligence/virus-worm-diffs.html

[17] LONG, Johnny *Google hacking* Vyd. 1. Brno: Zoner Press, 2005, 472 s. Encyklopedie Zoner Press ISBN 8086815315

[18] NOVÁK, Lukáš *Google Hacking [online]*, [cit. 2015-04-9]. Dostupné z: http://www.lukasnovak.net/skolni-prace/kib-google-hacking/

[19] PODERMANSKI Tomáš, GRÉGR Matěj *Bezpečné IPv6 : směrovač se hlásí [online]*, [cit. 2015-04-9]. Dostupné z: http://www.root.cz/clanky/bezpecne-ipv6-smerovac-se-hlasi/

[20] PODERMANSKI Tomáš, GRÉGR Matěj *Bezpečné IPv6: vícehlavý útočník [online]*, [cit. 2015-04-9]. Dostupné z: http://www.root.cz/clanky/bezpecne-ipv6-vicehlavy-utocnik/

[21] PODERMANSKI Tomáš, GRÉGR Matěj *Bezpečné IPv6: trable s multicastem [online]*, [cit. 2015-04-9]. Dostupné z: http://www.root.cz/clanky/bezpecne-ipv6-trable-s-multicastem/

[22] PODERMANSKI Tomáš, GRÉGR Matěj *Bezpečné IPv6: když dojde keš [online]*, [cit. 2015-04-9]. Dostupné z: http://www.root.cz/clanky/bezpecne-ipv6-kdyz-dojde-kes/

[23] MAURO Douglas R., SCHMIDT Kevin J. *Essential SNMP, Second Edition* Vyd. 2. Beijing: O'Reilly, 2005, ISBN 0596008406

[24] iREASONING.COM *MIB Browser [online]*, [cit. 2015-04-9]. Dostupné z: http://ireasoning.com/mibbrowser.shtml

[25] CISCO.COM *SNMP Object Navigator [online]*, [cit. 2015-04-9]. Dostupné z: http://tools.cisco.com/Support/SNMP/do/BrowseOID.do

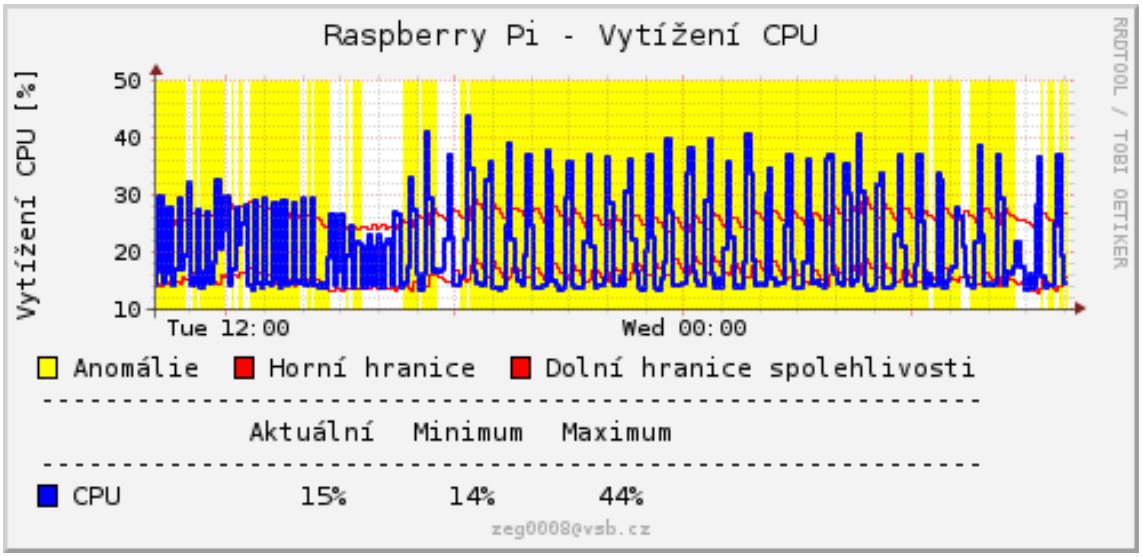
[26] SZMIT, Maciej *SNORT.AD [online]*, [cit. 2015-04-9]. Dostupné z: http://anomalydetection.info/

[27] OETIKER, Tobias *RRDtool [online]*, [cit. 2015-04-9]. Dostupné z: https://oss.oetiker.ch/rrdtool/doc/index.en.html

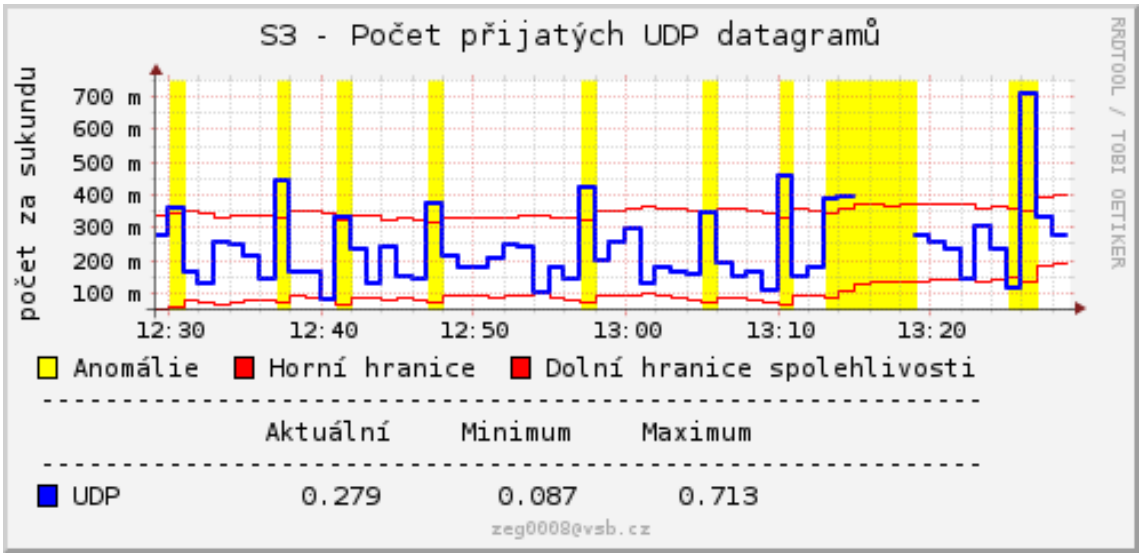
[28] OETIKER, Tobias *Aberrant Behavior Detection [online]*, [cit. 2015-04-9]. Dostupné z: http://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html

[29] GNS3.COM *[online]*, [cit. 2015-04-9]. Dostupné z: http://www.gns3.com/

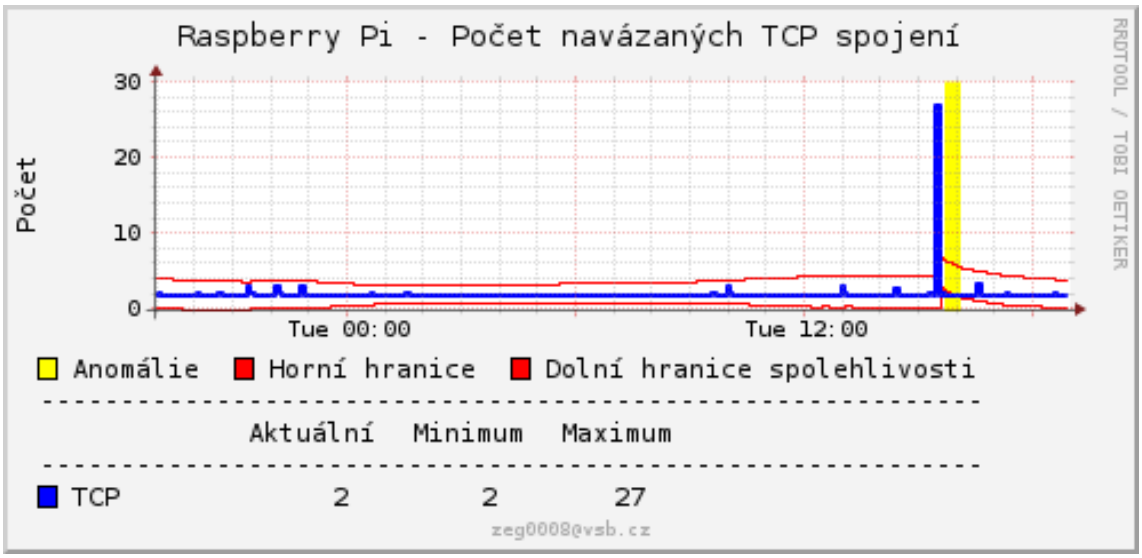
A Grafy z testování



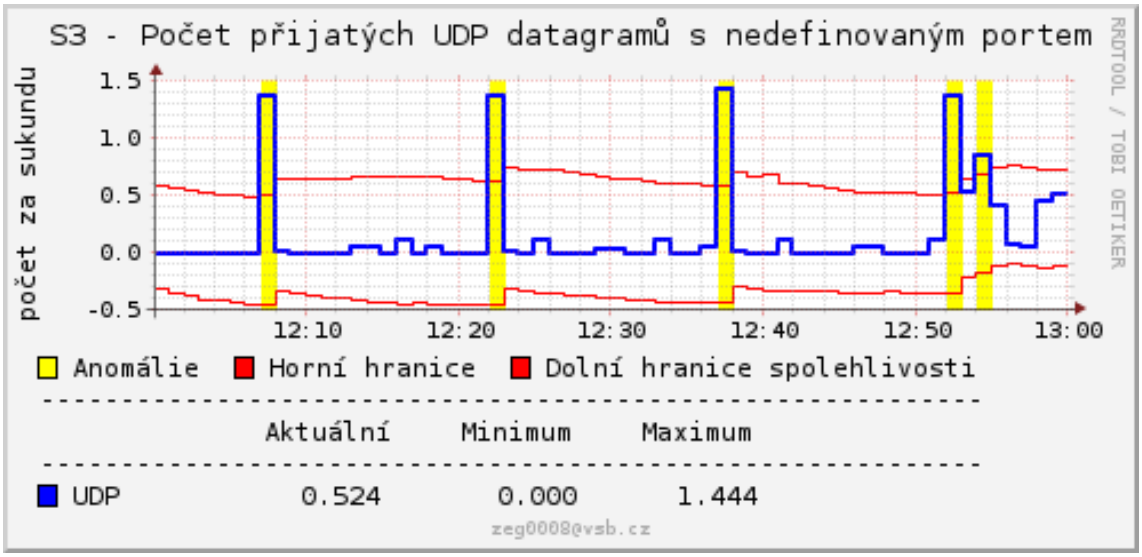
Obrázek 28: Nejdéle zaznamenaný útok



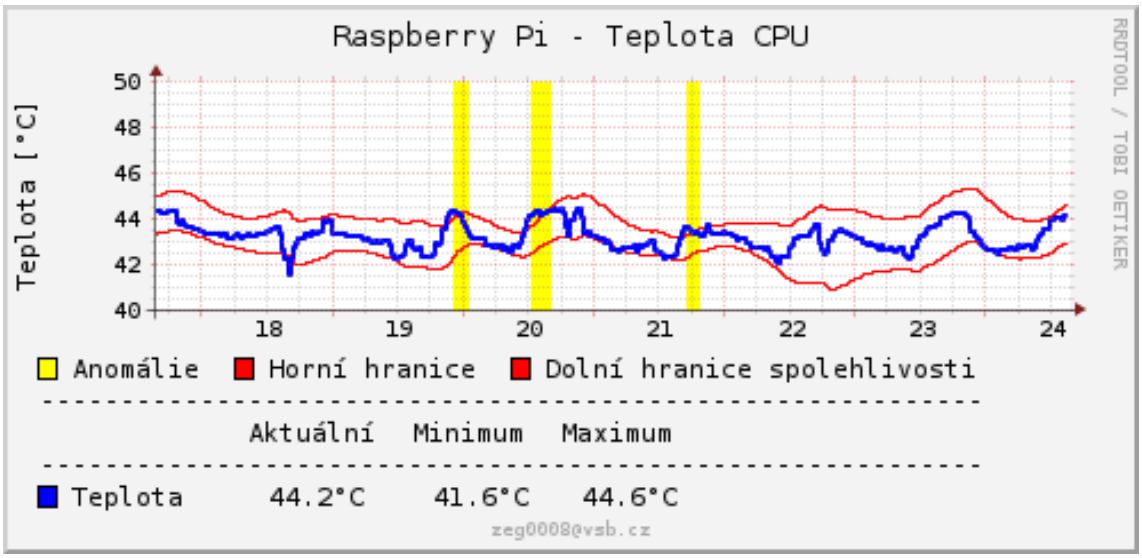
Obrázek 29: Systém detekoval i neočekávaný výpadek spojení



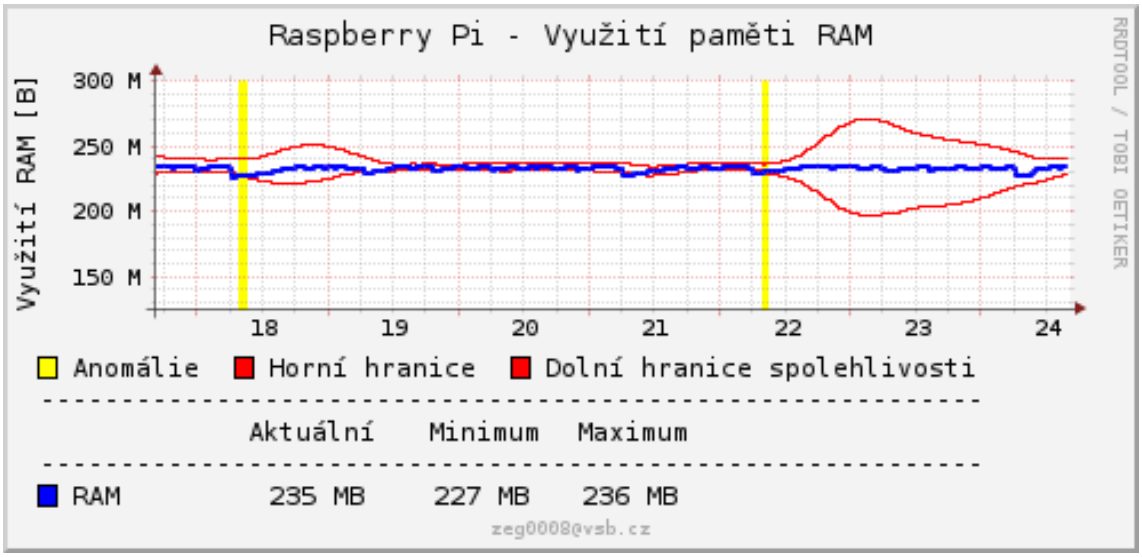
Obrázek 30: Detekce přístupu - zálohování dat



Obrázek 31: Detekce anomálií u protokolu UDP



Obrázek 32: Týdenní teplota CPU na Raspberry Pi



Obrázek 33: Týdenní vytížení paměti RAM na Raspberry Pi

B Obsah CD

Součástí diplomové práce je CD.

Obsah přiloženého CD:

1. text diplomové práce ve formátu PDF
2. složka IDS
    - složka grafy - výsledky z testování systému
    - složka skripty - skripty pro detekci anomálií napsané v Perlu
    - složka web - šablona pro webový server